43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science

FSTTCS 2023, December 18–20, 2023, IIIT Hyderabad, Telangana, India

Edited by

Patricia Bouyer
Srikanth Srinivasan

LIPICS

LIPIcs – Vol. 284 – FSTTCS 2023 www.dagstuhl.de/lipics

Editors

Patricia Bouyer
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, Gif-sur-Yvette, France
bouyer@lmf.cnrs.fr

Srikanth Srinivasan
Aarhus University, Denmark
University of Copenhagen, Denmark
srikanth@cs.au.dk

ACM Classification 2012
Theory of computation; Computing methodologies; Software and its engineering

ISBN 978-3-95977-304-1

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/978-3-95977-304-1.

Publication date
December, 2023

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.FSTTCS.2023.0

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)
- 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)
- Pierre Senellart (ENS, Université PSL, Paris, FR)

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface
Patricia Bouyer and Srikanth Srinivasan 0:ix

Programme Committee
..... 0:xi

List of External Reviewers: Track A
..... 0:xiii

List of External Reviewers: Track B
..... 0:xv

Invited Talks

Reachability Games and Friends: A Journey Through the Lens of Memory and Complexity
Thomas Brihaye, Aline Goeminne, James C. A. Main, and Mickael Randour 1:1–1:26

On Measuring Average Case Complexity via Sum-Of-Squares Degree
Prasad Raghavendra 2:1–2:1

Computational and Information-Theoretic Questions from Causal Inference
Leonard J. Schulman 3:1–3:1

From Concept Learning to SAT-Based Invariant Inference
Sharon Shoham 4:1–4:1

Algorithms in the Presence of Biased Inputs
Nisheeth K. Vishnoi 5:1–5:2

Regular Papers

Online Facility Location with Weights and Congestion
Arghya Chakraborty and Rahul Vaze 6:1–6:22

An Optimal Algorithm for Sorting in Trees
Jishnu Roychoudhury and Jatin Yadav 7:1–7:14

Parameterized Complexity of Biclique Contraction and Balanced Biclique Contraction
R. Krithika, V. K. Kutty Malu, Roohani Sharma, and Prafullkumar Tale 8:1–8:18

Towards Identity Testing for Sums of Products of Read-Once and Multilinear Bounded-Read Formulae
Pranav Bisht, Nikhil Gupta, and Ilya Volkovich 9:1–9:23

Bandwidth of Timed Automata: 3 Classes
Eugene Asarin, Aldric Degorre, Cătălin Dima, and Bernardo Jacobo Inclán 10:1–10:17

Monotone Classes Beyond VNP
Prerona Chatterjee, Kshitij Gajjar, and Anamay Tengse 11:1–11:23

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Approximate Maximum Rank Aggregation: Beyond the Worst-Case
Yan Hong Yao Alvin and Diptarka Chakraborty 12:1–12:21

Reinforcement Planning for Effective ε-Optimal Policies in Dense Time with Discontinuities
Léo Henry, Blaise Genest, and Alexandre Drewery 13:1–13:18

Parameterized Complexity of Incomplete Connected Fair Division
Harmender Gahlawat and Meirav Zehavi 14:1–14:18

Regular Separators for VASS Coverability Languages
Chris Köcher and Georg Zetzsche 15:1–15:19

Acyclic Petri and Workflow Nets with Resets
Dmitry Chistikov, Wojciech Czerwiński, Piotr Hofman, Filip Mazowiecki, and Henry Sinclair-Banks 16:1–16:18

Robust Positivity Problems for Linear Recurrence Sequences: The Frontiers of Decidability for Explicitly Given Neighbourhoods
Mihir Vahanwala 17:1–17:21

Nash Equilibria of Two-Player Matrix Games Repeated Until Collision
Aniket Murhekar and Eklavya Sharma 18:1–18:17

Synchronized CTL over One-Counter Automata
Shaull Almagor, Daniel Assa, and Udi Boker 19:1–19:22

A Class of Rational Trace Relations Closed Under Composition
Dietrich Kuske 20:1–20:20

Towards a Practical, Budget-Oblivious Algorithm for the Adwords Problem Under Small Bids
Vijay V. Vazirani 21:1–21:14

Languages Given by Finite Automata over the Unary Alphabet
Wojciech Czerwiński, Maciej Dębski, Tomasz Gogasz, Gordon Hoi, Sanjay Jain, Michał Skrzypczak, Frank Stephan, and Christopher Tan 22:1–22:20

Bounded Simultaneous Messages
Andrej Bogdanov, Krishnamoorthy Dinesh, Yuval Filmus, Yuval Ishai, Avi Kaplan, and Sruthi Sekar 23:1–23:17

Interval Selection in Data Streams: Weighted Intervals and the Insertion-Deletion Setting
Jacques Dark, Adithya Diddapur, and Christian Konrad 24:1–24:17

Listing 4-Cycles
Amir Abboud, Seri Khoury, Oree Leibowitz, and Ron Safier 25:1–25:16

Monotonicity Characterizations of Regular Languages
Yoav Feinstein and Orna Kupferman 26:1–26:19

Decision Tree Complexity Versus Block Sensitivity and Degree
Rahul Chugh, Supartha Podder, and Swagato Sanyal 27:1–27:23

FPT Approximations for Packing and Covering Problems Parameterized by Elimination Distance and Even Less
Tanmay Inamdar, Lawqueen Kanesh, Madhumita Kundu, M. S. Ramanujan, and Saket Saurabh 28:1–28:16

Leakage Resilience, Targeted Pseudorandom Generators, and Mild Derandomization of Arthur-Merlin Protocols
Dieter van Melkebeek and Nicollas Mocelin Sdroievski 29:1–29:22

Randomized and Quantum Query Complexities of Finding a King in a Tournament
Nikhil S. Mande, Manaswi Paraashar, and Nitin Saurabh 30:1–30:19

A Generalized Quantum Branching Program
Debajyoti Bera and Tharrmashastha SAPV 31:1–31:21

Tight Bounds for the Randomized and Quantum Communication Complexities of Equality with Small Error
Olivier Lalonde, Nikhil S. Mande, and Ronald de Wolf 32:1–32:18

Revisiting Mulmuley: Simple Proof That Maxflow Is Not in the Algebraic Version of *NC*
Ulysse Léchine 33:1–33:12

Solving Odd-Fair Parity Games
Irmak Sağlam and Anne-Kathrin Schmuck 34:1–34:24

New Lower Bounds for Reachability in Vector Addition Systems
Wojciech Czerwiński, Ismaël Jecker, Sławomir Lasota, Jérôme Leroux, and Łukasz Orlikowski 35:1–35:22

Approximately Interpolating Between Uniformly and Non-Uniformly Polynomial Kernels
Akanksha Agrawal and M. S. Ramanujan 36:1–36:17

Hardness of Learning Boolean Functions from Label Proportions
Venkatesan Guruswami and Rishi Saket 37:1–37:15

Dependency Schemes in CDCL-Based QBF Solving: A Proof-Theoretic Study
Abhimanyu Choudhury and Meena Mahajan 38:1–38:18

Weighted One-Deterministic-Counter Automata
Prince Mathew, Vincent Penelle, Prakash Saivasan, and A.V. Sreejith 39:1–39:23

Comparing Infinitary Systems for Linear Logic with Fixed Points
Anupam Das, Abhishek De, and Alexis Saurin 40:1–40:17

Constraint LTL with Remote Access
Ashwin Bhaskar and M. Praveen 41:1–41:21

Counter Machines with Infrequent Reversals
Alain Finkel, Shankara Narayanan Krishna, Khushraj Madnani, Rupak Majumdar, and Georg Zetzsche 42:1–42:17

Perfect Matchings and Popularity in the Many-To-Many Setting
Telikepalli Kavitha and Kazuhisa Makino 43:1–43:16

Preface

This volume contains the proceedings of the 43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023). The conference was held during the period December 18 – 20, 2023 on the campus of IIIT Hyderabad in Hyderabad, India. The conference was held as an in-person event.

The conference has two tracks: Track A focusing on algorithms, complexity and related issues, and Track B focusing on logic, automata and other formal method aspects of computer science. Each track had its own Program Committee (PC) and chair (Srikanth Srinivasan for Track A and Patricia Bouyer for Track B). This volume constitutes the joint proceedings of the two tracks, published in the LIPIcs series under a Creative Commons license, with free online access for all.

The conference comprised of 5 invited talks, 21 contributed talks in Track A, and 17 in Track B. This volume contains all the contributed papers from the two tracks, short abstracts of four of the invited talks as well as a longer paper accompanying one of the invited talks. We thank all the authors who submitted their papers to FSTTCS 2023. We are especially grateful to the PC members for their tireless work, and all the external reviewers for their expert opinion in the form of timely reviews.

We thank all the invited speakers for accepting our invitation: Thomas Brihaye (Université de Mons), Prasad Raghavendra (University of California at Berkeley), Leonard Schulman (California Institute of Technology), Sharon Shoham (Tel Aviv University), and Nisheeth Vishnoi (Yale University).

The main conference was accompanied by three workshops: Algorithmic Frontiers of Fairness (held on December 16 and 17, 2023 and organized by Siddharth Barman, Umang Bhaskar and Sujit Gujar), Research Highlights in Programming Languages (held during the period December 18-20, 2023 and organized by Deepak D'Souza, Uday Khedker, Kumar Madhukar, Kartik Nagar, Ganesan Ramalingam, Aseem Rastogi, Abhik Roychoudhury and Abhisekh Sankaran), and Spectral methods in Algorithms (held on December 21 and 22, 2023 and organized by Akash Kumar, Anand Louis and Piyush Srivastava).

We are indebted to the organizing committee for making all the necessary arrangements for the conference: the co-chairs Suryajith Chillara and Girish Varma (IIIT Hyderabad), Siddhartha Das (IIIT Hyderabad), Gowtham R. Kurri (IIIT Hyderabad), Grace Nalini (Outreach department at IIIT Hyderabad), Chandrashekhar Sahasrabudhe (Persistent systems/ACM India), and Abhijat Vichare (ACM India).

We thank S.P. Suresh (CMI, Chennai) for maintaining the conference web page. We thank Michael Wagner and the friendly staff at Dagstuhl LIPICs for helping us put together the proceedings. Finally, we thank the members of the Steering Committee, especially Amit Kumar, and the PC Chairs from FSTTCS 2022 (Venkat Guruswami and Anuj Dawar), for providing pertinent information and valuable advice about various aspects of the conference.

Patricia Bouyer and Srikanth Srinivasan
October 2023

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Programme Committee

Track A

- Siddharth Barman (The Indian Institute of Science, Bengaluru, India)
- Arnab Bhattacharyya (National University of Singapore, Singapore)
- Parinya Chalermsook (Aalto University, Finland)
- Sivakanth Gopi (Microsoft Research Redmond, USA)
- Karthik C.S. (Rutgers University, USA)
- Venkata Koppula (IIT Delhi, India)
- Rucha Kulkarni (University of Illinois, Urbana-Champaign, USA)
- Mrinal Kumar (TIFR, Mumbai, India)
- Paloma Lima (IT University of Copenhagen, Denmark)
- Meena Mahajan (The Institute of Mathematical Sciences, Chennai, India)
- Nikhil Mande (University of Liverpool, UK)
- Neeldhara Misra (IIT Gandhinagar, India)
- Sagnik Mukhopadhyay (University of Sheffield, UK)
- Meghana Nasre (IIT Madras, India)
- Noga Ron-Zewi (University of Haifa, Israel)
- Chris Schwiegelshohn (Aarhus University, Denmark)
- Srikanth Srinivasan (Aarhus University and University of Copenhagen, Denmark) – Track A chair

Track B

- Aiswarya C. (Chennai Mathematical Institute, India)
- Guy Avni (University of Haifa, Israel)
- Francesco Belardinelli (Imperial College, UK)
- Dietmar Berwanger (CNRS, LMF, France)
- Patricia Bouyer (CNRS, LMF, France) – Track B chair
- Arnaud Carayol (Université Gustave Eiffel, LIGM, France)
- Supratik Chakraborty (IIT Bombay, India)
- Lorenzo Clemente (University of Warsaw, Poland)
- Ugo Dal Lago (Università di Bologna, Italy)
- P. Madhusudan (University of Illinois, Urbana-Champaign, USA)
- Pavithra Prabhakar (Kansas State University, USA)
- Prakash Saivasan (The Institute of Mathematical Science, Chennai, India)
- Abhisekh Sankaran (Tata Consultancy Services Research, Pune, India)
- B. Srivathsan (Chennai Mathematical Institute, India)
- Nathalie Sznajder (Sorbonne Université, LIP6, France)
- Tony Tan (University of Liverpool, UK)

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of External Reviewers: Track A

Marek Adamczyk
Thomas Archbold
Vikrant Ashvinkumar
Nikhil Balaji
Aritra Banik
Aniket Basu Roy
Benjamin Aram Berendsohn
Amey Bhangale
Karl Bringmann
Jakob Burkhardt
Diptarka Chakraborty
Yi-Jun Chang
Keerti Choudhary
Yogesh Dahiya
Veera Venkata Narayana Dogiparthy
Dean Doron
Aditi Dudeja
Alireza Farhadi
Karl Fehrs
Ameet Gadekar
Prantar Ghosh
Sumanta Ghosh
Uma Girish
Petr Golovach
Zeyu Guo
Chetan Gupta
Rohit Gurjar
Thekla Hamm
Kaave Hosseini
Siddhartha Jain
Debajyoti Kar
Kaspar Kasche
Manas Jyoti Kashyop
Kamyar Khodamoradi
Alexander Knop
Caleb Koch
Balagopal Komarath
Sajin Koroth
Anand Krishna
Ariel Kulik
Girija Limaye
Allen Liu
Yanyi Liu
Harshil M.
Rahul Madhavan
Arnab Maiti
Rogers Mathew
Ian Mertz
Pranabendu Misra
Rajat Mittal
Matthias Mnich
Laure Morelle
Saraswati Nanoti
Shivika Narang
Prajakta Nimbhorkar
Manaswi Paraashar
Subhasree Patro
Rameesh Paul
Supartha Podder
Nidhi Purohit
Pablo Pérez-Lantero
Venkatesh Raman
C Ramya
Nidhi Rathi
Mahesh S. R.
Ron Safier
Ramprasad Saptharishi
Suneel Sarswat
Aditi Sethia
Sudarshan Shyam
Joachim Spoerhase
Karteek Sreenivasaiah
Adarsh Srinivasan
Aleksa Stankovic
Prafullkumar Tale
Anamay Tengse
Rohit Vaish
Daniel Vaz
Karnati Venkata Naga Sreenivasulu
Sebastian Wild
Michal Wlodarczyk
Weiqiang Yuan

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of External Reviewers: Track B

Paolo Baldan
Marie-Pierre Béal
Benjamin Böhm
Titouan Carette
Soham Chakraborty
Jahid Chowdhury Choton
Ramit Das
Laure Daviaud
Stéphane Demri
Antonio Di Stasio
Catalin Dima
Léo Exibard
Florian Furbach
Francesco Gavazzo
Shibashis Guha
Lipsy Gupta
Stefan Göller
Peter Habermehl
Matthew Hague
Piotr Hofman
Lukáš Holík
Uday Khedker
Florent Koechlin
Denis Kuperberg
Martin Lange
Christof Löding
Kumar Madhukar
Alexandre Mansard
Filip Mazowiecki
Hernan Melgratti
Pranabendu Misra
Benjamin Monmege
Anuj More
Diganta Mukhopadhyay
Madhavan Mukund
Aniket Murhekar
Muhammad Najib
K. Narayan Kumar
Luca Padovani
Paweł Parys
Vincent Penelle
Ramchandra Phawade
Paolo Pistone
Chris Pollett
M. Praveen
Tephilla Prince
R. Ramanujam
Susanna Rezende
Sparsa Roychowdhury
Krishna S
Suman Sadhukhan
Arnaud Sangnier
N. Ege Saraç
Sylvain Schmitz
Anindya Sen
Olivier Serre
Sunil Simon
Sathyawageeswar Subramanian
Chana Weil-Kennedy
Djordje Zikelic

Reachability Games and Friends: A Journey Through the Lens of Memory and Complexity

Thomas Brihaye
UMONS – Université de Mons, Belgium

Aline Goeminne
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

James C. A. Main
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Mickael Randour
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Abstract

Reachability objectives are arguably the most basic ones in the theory of games on graphs (and beyond). But far from being bland, they constitute the cornerstone of this field. Reachability is everywhere, as are the tools we use to reason about it. In this invited contribution, we take the reader on a journey through a zoo of models that have reachability objectives at their core. Our goal is to illustrate how model complexity impacts the complexity of strategies needed to play optimally in the corresponding games and computational complexity.

2012 ACM Subject Classification Software and its engineering → Formal methods; Theory of computation → Logic and verification; Theory of computation → Solution concepts in game theory

Keywords and phrases Games on graphs, reachability, finite-memory strategies, complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.1

Category Invited Talk

Funding This work has been supported by the Fonds de la Recherche Scientifique – FNRS under Grant n° T.0027.21 (PDR RatBeCoSi) and Grant n° T.0188.23 (PDR ControlleRS).
Aline Goeminne: Postdoctoral Researcher of the Fonds de la Recherche Scientifique – FNRS.
James C. A. Main: Research Fellow of the Fonds de la Recherche Scientifique – FNRS, member of the TRAIL Institute.
Mickael Randour: Research Associate of the Fonds de la Recherche Scientifique – FNRS, member of the TRAIL Institute.

Acknowledgements The authors are grateful to Patricia Bouyer-Decitre for carefully reading through previous versions of this paper and providing valuable comments.

1 Introduction

Games on Graphs. We consider infinite-duration multi-player turn-based games played on graphs [34], often called *arenas*. The set of vertices of the graph is partitioned between the players. Players interact by moving a pebble from vertex to vertex, ad infinitum, following edges of the graph. The game starts in a given vertex, and the owner of the current vertex decides where to send the pebble next. The infinite path thereby created is called a *play*.

Objectives and Strategies. Each player has an *objective function*, which is essentially a measure of the utility of plays: players aim to optimise their utility. Players choose their actions (i.e., where to move the pebble) according to a *strategy*.

© Thomas Brihaye, Aline Goeminne, James C. A. Main, and Mickael Randour;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 1; pp. 1:1–1:26
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A key question in the field is to understand how complex strategies need to be for players to play optimally, when possible. In full generality, strategies may look at the current history – which may be arbitrarily long as we consider infinite-duration games – to pick an action. They may also use randomness, that is, prescribe a probability distribution over possible actions. However, in the simplest settings, this is superfluous. In particular, players may often resort to *pure memoryless* strategies, which simply associate a single action to each vertex, independently of what happened before.

When Are Complex Strategies Needed? Games on graphs and related models, such as Markov decision processes or stochastic games (both involving probabilistic arenas), have been studied extensively for decades, for a plethora of objectives [34]. An interesting line of work tries to understand, classify, and characterise broad classes of games and objectives for which strategies of restricted classes of complexity suffice to play optimally. One can notably mention [42] for (pure) memoryless (resp. [11] for finite-memory) strategies in two-player zero-sum games, [43] for (pure) memoryless (resp. [12] for finite-memory) strategies in two-player zero-sum *stochastic* games, [53] for finite-memory strategies when considering *combinations of objectives*, or [63] for *multi-player* games.

The Different Flavours of Complexity. As discussed above, the recent years have seen a surge in research to understand when (specific classes of) complex strategies are needed. Yet, one may take a step back and ask what does it really mean for a strategy to be complex? And what is the role of this complexity? With respect to memory, for example, we know that different models co-exist: see, e.g., the discussion on chromatic vs. chaotic memory in a recent survey [13]. Similarly, the various ways one may allow strategies to use randomness has a huge impact on the (expressive) power of strategies [56].

Formalising the nature of complexity is not an easy feat, and subject to almost-philosophical debates. What we aim to do in this work is a small step along the way: we will take a walk together among various game models and objectives, of (arguably) increasing complexity, and try to understand how this complexity percolates through the corresponding strategies, as well as on the computational standpoint. It is easy to get lost in the zoo of models and objectives that exist in the literature; to avoid that, we will focus on the familiar yet surprisingly interesting *reachability* objectives.

Reachability. In its simplest form, a reachability objective [34] simply asks to visit once any vertex belonging to a given target set. This objective is often considered to be the most elementary, and the main tool to solve reachability games, *attractors*, has made its way in virtually all algorithms on games on graphs. That is because, in one way or another, strategies for most game objectives do involve going from a point A to a point B as part of their inner mechanisms. This small part is exactly a reachability objective.

One may also note that the prevalence of reachability is not limited to games on graphs. Consider for example the effort put in understanding the reachability problem in Petri nets [54, 28], or in more distant models, the Skolem problem [74].

Outline. Our journey will go through reachability, generalised reachability (i.e., multiple target sets), shortest path (i.e., minimise the costs up to the first visit), repeated reachability (aka Büchi) objectives... and consider various models such as multi-player or stochastic games. As stated before, our goal is to illustrate how and when complexity arises, and what is the role of that complexity in strategies. To do so, we will work our way through increasingly complex models and objectives, grouped with respect to the resulting complexity of the corresponding strategies.

Our presentation is high-level and far from exhaustive: we do not aim to provide a fully-filled table of all possible combinations of models and objectives. Instead, we pick and choose particular settings that exemplify interesting phenomena. As such, this work should not be seen as a definitive overview of reachability games, but as a particular (and hopefully interesting) path through this crowded zoo.

2 Preliminaries

Notation. We let $\mathbb{R}$ (resp. $\mathbb{N}$, $\mathbb{Z}$) denote the set of real (resp. non-negative integer, integer) numbers and $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty, -\infty\}$ (resp. $\overline{\mathbb{N}} = \mathbb{N} \cup \{+\infty\}$, $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{+\infty, -\infty\}$). For any integer $k \geq 1$, we let $[\![k]\!] = \{1, \dots, k\}$.

Games. Let (V, E) be a directed graph where V is a finite set of vertices and $E \subseteq V \times V$ is a set of edges. An (n-player) *arena* (on (V, E)) is a tuple $\mathcal{A} = ((V_i)_{i \in [\![n]\!]}, E)$ where $[\![n]\!]$ is a set of n players and $(V_i)_{i \in [\![n]\!]}$ is a partition of V (between the different players). We assume that our arena is deadlock-free, i.e., for all $v \in V$, there exists $v' \in V$ such that $(v, v') \in E$. For $i \in [\![n]\!]$, we write $\mathcal{P}_i$ to denote the ith player.

Players take turns choosing the next vertex depending on which player controls the current vertex. This forms an infinite path in the arena called a *play*. Formally, a play in $\mathcal{A}$ is an infinite sequence of vertices $\rho = \rho_0 \rho_1 \dots$ such that for all $\ell \in \mathbb{N}$, $(\rho_\ell, \rho_{\ell+1}) \in E$. A history is a finite prefix of a play $h = h_0 \dots h_\ell$; we denote h_ℓ by $\mathsf{last}(h)$. The set of plays (resp. histories) is denoted by Plays (resp. Hist) and Hist_i is the set of histories ending in a vertex of $\mathcal{P}_i$. For $v \in V$, $\mathrm{Plays}(v)$, $\mathrm{Hist}(v)$, $\mathrm{Hist}_i(v)$ denote respectively the sets of plays, histories and histories ending in a vertex of $\mathcal{P}_i$ that begin in v. In an arena, it is common to fix an initial vertex $v_0 \in V$ and consider only plays and histories starting in v_0.

A (n-player) *game* $\mathcal{G} = (\mathcal{A}, (\mathrm{Obj}_i)_{i \in [\![n]\!]})$ is an arena equipped with a profile of *objective functions* $\mathrm{Obj} = (\mathrm{Obj}_i)_{i \in [\![n]\!]}$. For $i \in [\![n]\!]$, $\mathrm{Obj}_i : \mathrm{Plays} \to \overline{\mathbb{R}}$ measures the utility of each play of $\mathcal{A}$; $\mathcal{P}_i$ aims to optimise Obj_i. If $\mathcal{P}_i$ wants to maximise (resp. minimise) Obj_i, we say that Obj_i is a *gain* (resp. *cost*) *function* and we denote it by Gain_i (resp. Cost_i). In what follows, unless otherwise stated, we assume that the considered objective functions are gain functions.

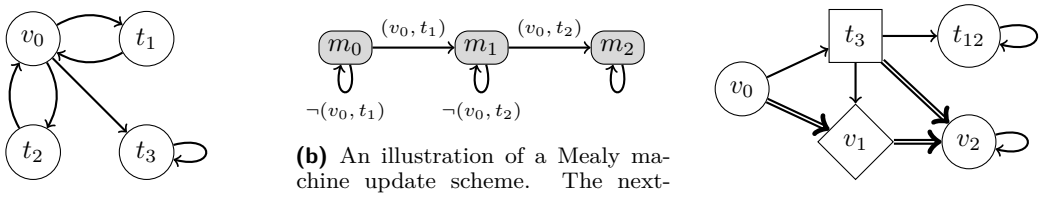
A two-player game $\mathcal{G} = (\mathcal{A}, (\mathrm{Obj}_1, \mathrm{Obj}_2))$ is *zero-sum* if, for all $\rho \in \mathrm{Plays}$, we have that[1] $\mathrm{Obj}_1(\rho) + \mathrm{Obj}_2(\rho) = 0$. Zero-sum games are often abusively denoted $\mathcal{G} = (\mathcal{A}, \mathrm{Obj}_1)$, as Obj_2 can be deduced from Obj_1. Given a play ρ, whenever Obj_1 represents a Boolean objective, i.e., Obj_1 has codomain $\{0, 1\}$, we say that $\mathcal{P}_1$ wins (resp. loses) if $\mathrm{Obj}_1(\rho) = 1$ (resp. 0). The Boolean designation is abusive for $\mathcal{P}_2$, as their gains range in $\{0, -1\}$.

Reachability Objective Functions. We now define the (generalised) reachability objective functions. Quantitative variants are discussed and defined in Section 3.3.

Let $\mathrm{T} \subseteq V$ be a target (set), the *reachability objective (for* T*)* consists in visiting T. Formally, we have that $\mathsf{Reach}[\mathrm{T}](\rho) = 1$, if there exists a position $\rho_\ell \in \mathrm{T}$, otherwise $\mathsf{Reach}[\mathrm{T}](\rho) = 0$. Let $\mathrm{T}_1, \dots \mathrm{T}_k \subseteq V$ be target sets. The *generalised reachability objective (for* $\mathrm{T}_1, \dots \mathrm{T}_k$*)* consists in visiting all the sets T_j in any order. Formally we have that $\mathsf{GReach}[\mathrm{T}_1, \dots, \mathrm{T}_k](\rho) = \prod_{j=1}^{k} \mathsf{Reach}[\mathrm{T}_j](\rho)$.

▶ Remark 1. We refer to games of the form $\mathcal{G} = (\mathcal{A}, (\mathsf{Reach}[\mathrm{T}_i])_{i \in [\![n]\!]})$ as *reachability games*. In the two-player zero-sum case, we speak of zero-sum reachability games if $\mathcal{P}_1$ has a reachability objective, although $\mathcal{P}_2$ does not have a reachability objective. We proceed similarly for the other considered objective functions.

[1] If Obj_1 can take infinite values, we apply the convention $(+\infty) + (-\infty) = 0$.



(a) A one-player generalised reachability game where $\mathrm{T}_j = \{t_j\}$ for all $j \in [\![3]\!]$.

(b) An illustration of a Mealy machine update scheme. The next-move function of the machine is partly given by $\mathsf{nxt}(m_0, v_0) = t_1$, $\mathsf{nxt}(m_1, v_0) = t_2$ and $\mathsf{nxt}(m_2, v_0) = t_3$.

(c) A three-player reachability game with an NE that is not an SPE. Double arrows denote a (memoryless) strategy profile.

Figure 1 A one-player game (Figure 1a) with a Mealy machine encoding a strategy that is winning from v_0 (Figure 1b) and a game illustrating the difference between NEs and SPEs (Figure 1c).

Strategies. A *(pure) strategy* σ_i of $\mathcal{P}_i$, $i \in [\![n]\!]$, provides the next action of $\mathcal{P}_i$ based on what occurred previously. Formally, $\sigma_i \colon \mathrm{Hist}_i \to V$ is a function such that for all $h = h_0 \dots h_\ell \in \mathrm{Hist}_i$, $(h_\ell, \sigma_i(h)) \in E$. We denote by Σ_i the set of strategies of $\mathcal{P}_i$. We say that a play ρ is *played according to* σ_i if, whenever $\rho_\ell \in V_i$, we have that $\sigma_i(\rho_0 \dots \rho_\ell) = \rho_{\ell+1}$. We differentiate two classes of strategies: *memoryless* strategies and *finite-memory* strategies. A *memoryless* strategy σ_i only depends on the last vertex of the history, i.e., for all $h, h' \in \mathrm{Hist}_i$, if $\mathsf{last}(h) = \mathsf{last}(h')$, then $\sigma_i(h) = \sigma_i(h')$.

A strategy is *finite-memory* if it can be encoded by a finite-state Mealy machine. Formally, such a machine (for $\mathcal{P}_i$) is a tuple $\mathfrak{M} = (\mathrm{M}, m_0, \mathsf{up}, \mathsf{nxt})$ where M is a finite set of states, m_0 is an initial memory state, $\mathsf{up} \colon \mathrm{M} \times E \to \mathrm{M}$ is a memory update function and $\mathsf{nxt} \colon \mathrm{M} \times V_i \to V$ is a next-move function. Intuitively, the strategy encoded by a Mealy machine prescribes moves through the nxt function based on the current memory state and game vertex, and the memory state is updated at each round via the up function based on the traversed edge. In the following, the size of a memory for a strategy refers to an upper bound on the number of states of some Mealy machine that encodes the strategy.

Mealy machines combine memoryless strategies: a next-move function assigns a memoryless strategy to each memory state. In particular, memoryless strategies are finite-memory strategies that can be represented by Mealy machines with a single memory state.

▶ Remark 2. In some presentations, the update function of a Mealy machine is defined using vertices rather than edges. Memory bounds for strategies may vary depending on the model, e.g., when the objective function depends on edge-specific information (e.g., the shortest path games of Section 4.2). The vertex-update model is less compact in general.

A *strategy profile* $\sigma = (\sigma_i)_{i \in [\![n]\!]}$ assigns a strategy to each player. To highlight the role of $\mathcal{P}_i$, we sometimes write $\sigma = (\sigma_i, \sigma_{-i})$, where σ_{-i} denotes the strategy profile of the players other than $\mathcal{P}_i$. Once a strategy profile σ is fixed together with an initial vertex v_0, there exists a unique play starting from v_0 and played according to all strategies of σ. We denote this unique play by $\langle \sigma \rangle_{v_0}$ (or just $\langle \sigma \rangle$ when v_0 is clear from the context).

▶ **Example 3.** Let us consider the one-player game played on the arena given in Figure 1a with the objective $\mathsf{GReach}[\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3]$ where $\mathrm{T}_j = \{t_j\}$ for all $j \in [\![3]\!]$ and the initial vertex v_0. We provide in Figure 1b a Mealy machine $\mathfrak{M}$ that encodes a strategy σ_1 with a memory of size 3. The three memory states are m_0, m_1 and m_2 and the memory update function is illustrated as follows: *(i)* a label $(v, v') \in E$ (resp. $\neg(v, v')$) on a transition $(m, m') \in \mathrm{M} \times \mathrm{M}$ represents $\mathsf{up}(m, (v, v')) = m'$ (resp. $\mathsf{up}(m, e) = m'$ for all $e \in E \setminus \{(v, v')\}$) and *(ii)* an unlabelled transition (m, m') means $\mathsf{up}(m, e) = m'$ for all $e \in E$. We consider the next-move function partly defined as $\mathsf{nxt}(m_{j-1}, v_0) = t_j$ for all $j \in [\![3]\!]$. The outcome of the strategy σ_1 is obtained as follows: the play begins in v_0 and the initial memory state is m_0, thus the first move prescribed by σ_1 is $\mathsf{nxt}(m_0, v_0) = t_1$. Once transition (v_0, t_1) is crossed, the

memory state is updated to m_1. Since from t_1 the only outgoing edge enters v_0, the current vertex is v_0 again and the memory state does not change. This time, from v_0, we move to $\mathsf{nxt}(m_1, v_0) = t_2$ and the memory state is updated to m_2. Continuing this way results in the outcome $\langle \sigma_1 \rangle = v_0 t_1 v_0 t_2 v_0 t_3^\omega$ which visits all targets.

Solution Concepts. Let $\mathcal{G} = (\mathcal{A}, (\mathrm{Gain}_i)_{i \in [\![n]\!]})$ be a game with an initial vertex v_0.

Given $\alpha \in \overline{\mathbb{R}}$, a strategy σ_i is said to *ensure* α (from v_0) if for all strategy profiles of the opponents σ_{-i}, we have that $\mathrm{Gain}_i(\langle \sigma_i, \sigma_{-i} \rangle) \geq \alpha$. A strategy σ_i is said to be *optimal (for* $\mathcal{P}_i$*)* if there exists $\alpha \in \overline{\mathbb{R}}$ such that σ_i ensures α and for all $\beta > \alpha$, no strategy of $\mathcal{P}_i$ ensures β. If such an α exists, it is called the value of vertex v_0 in the two-player zero-sum game $\mathcal{G}_i = (((V_i, V \setminus V_i), E), \mathrm{Gain}_i)$, called the *coalition game* for $\mathcal{P}_i$, in which $\mathcal{P}_i$ is opposed to the coalition of the other players.

In a two-player zero-sum (or one-player) game with Boolean objective functions, we say that σ_1 is a *winning strategy (for* $\mathcal{P}_1$*)* if σ_1 ensures 1. In this case, σ_1 is necessarily optimal.

A strategy profile σ is a *Nash equilibrium* (NE) if no player has an incentive to deviate unilaterally in order to increase his gain, i.e., for all $i \in [\![n]\!]$, for all $\sigma'_i \in \Sigma_i$, $\mathrm{Gain}_i(\langle \sigma'_i, \sigma_{-i} \rangle) \leq \mathrm{Gain}_i(\langle \sigma \rangle)$. By essence, NEs do not take into account the sequential nature of games played on graphs. This implies existence of incredible threats in some NEs (see Example 4). The concept of *subgame perfect equilibrium* (SPE) proposes a classical remedy to this problem.

To formally define SPEs, we introduce the concept of subgame. Given a history $hv \in \mathrm{Hist}(v_0)$, the game $\mathcal{G}_{\restriction hv}$ with initial vertex v is called a *subgame* of $\mathcal{G}$ and is such that $\mathcal{G}_{\restriction hv} = (\mathcal{A}, (\mathrm{Gain}_{i \restriction hv})_{i \in [\![n]\!]})$ with, for all $i \in [\![n]\!]$ and for all plays $\rho \in \mathrm{Plays}(v)$, $\mathrm{Gain}_{i \restriction hv}(\rho) = \mathrm{Gain}_i(h\rho)$. Given a strategy of $\mathcal{P}_i$ in $\mathcal{G}$, $\sigma_{i \restriction hv}$ denotes its restriction in the subgame $\mathcal{G}_{\restriction hv}$ and is defined by, for all histories $h' \in \mathrm{Hist}_i(v)$, $\sigma_{i \restriction hv}(h') = \sigma_i(hh')$.

A strategy profile σ is an SPE in $\mathcal{G}$, if for all histories $hv \in \mathrm{Hist}(v_0)$, $\sigma_{\restriction hv} = (\sigma_{i \restriction hv})_{i \in [\![n]\!]}$ is an NE in $\mathcal{G}_{\restriction hv}$. Notice that, in particular, an SPE is an NE.

▶ **Example 4.** Let us consider the three-player reachability game played on the arena depicted in Figure 1c with initial vertex v_0. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ control circle, square and diamond vertices respectively and their targets are $\mathrm{T}_1 = \mathrm{T}_2 = \{t_{12}\}$ and $\mathrm{T}_3 = \{t_3\}$. A (memoryless) strategy profile σ is depicted by double arrows. We can prove that σ is an NE such that no player visits his target. However, it is not an SPE: $\mathcal{P}_2$ has an incentive to change his strategy in the subgame $\mathcal{G}_{\restriction v_0 t_3}$ by visiting $t_{12} \in \mathrm{T}_2$ instead of v_2. We define another strategy profile τ such that *(i)* $\tau_1(v_0) = t_3$ and $\tau_1(v) = v$ if $v \in \{v_2, t_{12}\}$, *(ii)* $\tau_2(t_3) = t_{12}$ and *(iii)* $\tau_3(v_1) = v_2$. This strategy profile is an SPE (thus also an NE) such that all players reach their target.

Studied Problems. We now define decision problems that arise naturally in our setting. In a one-player or two-player zero-sum setting, we study how well a player can perform.

▶ **Problem 1** (Threshold-ensuring strategy existence (TSE) problem)**.** *Given a two-player zero-sum (or one-player) game* $\mathcal{G} = (\mathcal{A}, \mathrm{Gain}_1)$ *with initial vertex* v_0 *and a threshold* $\alpha \in \overline{\mathbb{R}}$*, does there exist a strategy* σ_1 *of* $\mathcal{P}_1$ *such that* σ_1 *ensures* α*?*

We say that a strategy *solves* the problem if it provides a positive answer to the problem. When studying a two-player zero-sum (or one-player) game with a Boolean objective function, by *solving the game*, we mean deciding the existence of a winning strategy for $\mathcal{P}_1$, i.e., deciding the TSE problem for a threshold $\alpha = 1$.

Regarding equilibria, SPEs (and therefore NEs) are guaranteed to exist in all games with ω-regular Boolean objectives [69]; these games subsume all Boolean objectives considered in Section 3. SPEs exist also in all games with continuous gain functions [40]. These games

include shortest path games (see Section 3.3) with positive weights on the edges through an appropriate transformation [17]. Furthermore, we have seen in Example 4 that an NE where no player visits his target may coexist with another NE such that all players visit their targets, i.e., there is not a unique equilibrium gain profile in a given game. Therefore, instead of studying the existence of equilibria, which is often guaranteed, the following decision problem is considered.

▶ **Problem 2** (Constrained equilibrium existence (CEE) problem). *Given an n-player game $\mathcal{G} = (\mathcal{A}, (\mathrm{Gain}_i)_{i \in [\![n]\!]})$ with initial vertex v_0 and a threshold tuple $\alpha \in \overline{\mathbb{R}}^n$, does there exist an equilibrium σ (NE or SPE) such that for all $i \in [\![n]\!]$, $\mathrm{Gain}_i(\langle\sigma\rangle) \geq \alpha_i$?*

We say that a strategy profile *solves* the CEE problem if it provides a positive answer to it. When studying an n-player game with Boolean objective functions, we only consider threshold tuples in $\{0,1\}^n$.

SAT, QBF and Generalised Reachability. We consider two classical problems: the *Boolean satisfiability (SAT) problem* and the *quantified Boolean formula (QBF) problem*, and their relations with generalised reachability. Although generalised reachability in the two-player zero-sum setting is discussed in Section 3.1.3, we present a hardness argument and an algorithm to solve such games below, as we refer to these throughout Section 3.1.

The QBF problem asks whether a fully quantified formula (in conjunctive normal form) is true or false. To be more specific, let us consider finitely many variables $x_1, \ldots, x_m$. A literal is a variable x_i or its negation $\neg x_i$. A clause C is a finite disjunction of literals. Let Q denote a quantifier, i.e., $Q \in \{\exists, \forall\}$. The QBF problem asks whether a formula of the form $\psi = Q_1 x_1\ Q_2 x_2\ \ldots\ Q_m x_m\ C_1 \wedge\ C_2 \wedge \ldots \wedge C_k$ is true.

It is classical to view the QBF problem as reachability in an And-Or graph (exponential in the size of the formula) [60]. Equivalently, one can also solve the QBF problem via a two-player zero-sum game (linear in the size of the formula) with a generalised reachability objective [35, 36]. We describe a generic construction to derive a graph from the formula ψ. For each quantifier Q_i and corresponding bound variable x_i, we introduce three vertices Q_i, x_i and $\neg x_i$ with edges from Q_i to the other two vertices and, if $i < m$, edges from x_i and $\neg x_i$ to Q_{i+1}. The resulting graph is represented in the small dashed rectangle $\mathbf{QBF_1}$ of Figure 2. We postpone the explanation of the whole figure $\mathbf{QBF_2}$; it is used in a specific reduction in Section 3.1.3. We say that vertex Q_i is an existential (resp. universal) vertex if Q_i is an existential (resp. universal) quantifier. For each clause C_ℓ of the formula, we introduce a target T_ℓ consisting of the vertices that are literals occurring in C_ℓ.

When reducing the QBF instance given by ψ to a two-player zero-sum generalised reachability game, we assign existential and universal vertices in the graph above to $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. The objective of $\mathcal{P}_1$ is given by $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$. Intuitively, the players, through their strategies, select the assignments of the variables bound to their quantifiers in the order of quantification and a clause holds if its matching target is visited. It follows that there is a winning strategy for $\mathcal{P}_1$ if and only if there is a way to assign existentially quantified variables such that, no matter how universally quantified variables are assigned, all clauses hold, i.e., if ψ holds [35, 36]. As the QBF problem is PSPACE-complete [60], the above reduction directly implies that solving two-player zero-sum games with generalised reachability objectives is PSPACE-hard.

One can prove that the problem of solving generalised reachability games is actually PSPACE-complete, following the lines of [35, 36]. It can be solved via an alternating polynomial time algorithm. The algorithm relies on the fact that there exists a winning strategy for $\mathcal{P}_1$ if and only if there is one that visits all targets within at most $k \cdot |V|$ steps. To

decide the winner, one simulates plays by choosing the next state existentially or universally depending on the player and until the previous bound is reached. Although the step counter only requires space logarithmic in $k \cdot |V|$, the algorithm does not use logarithmic space overall as it must keep track of the set of visited targets along a play.

The SAT problem, which is known to be NP-complete [26], can be seen as the restriction of the QBF problem where all the quantifiers are existential. The SAT problem can be reduced to solving a one-player generalised reachability game via the previously described reduction. It follows that solving one-player generalised reachability games is NP-hard [35, 36].

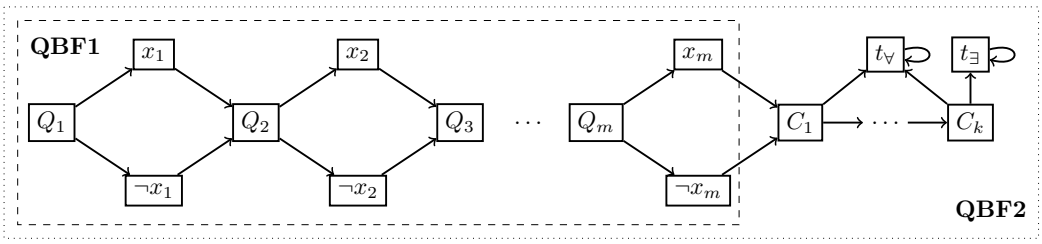


Figure 2 A graph derived from the formula $\psi = Q_1 x_1\ Q_2 x_2\ \ldots\ Q_m x_m\ C_1 \wedge\ C_2 \wedge \ldots \wedge C_k$. For $\ell \in [\![k]\!]$, all the literals of the clause C_ℓ are encoded in a target T_ℓ. For instance, given the clause $x_1 \vee \neg x_2 \vee x_3$, the associated target is $\{x_1, \neg x_2, x_3\}$.

3 The Structure and Properties of Memory

In this section, we overview strategies in games with reachability-related objectives. First, we consider Boolean objectives. We present (generalised) reachability in Section 3.1. In Section 3.2, we discuss games in which targets have to be visited infinitely often. Finally, in Section 3.3, we consider a quantitative variant of reachability, where we aim at reaching the target(s) with minimal cost. The problems in each section are presented in increasing order of complexity required by strategies that solve the problems. The complexity of a strategy refers not only to its memory size but also to the variety of the behaviours it encodes.

3.1 Boolean Reachability

We focus here on (generalised) reachability objectives. The section is divided into three parts. Section 3.1.1 is dedicated to problems where memoryless strategies suffice. In Section 3.1.2, we consider problems where it suffices to use a memory of polynomial size that encodes a (fixed) sequence representing the order in which targets are to be seen. Finally, in Section 3.1.3, we deal with problems where a memory that branches based mainly on the set of visited targets suffices. In this latter case, an exponential memory size may be required.

3.1.1 Memoryless Strategies

There are two simple situations where no memory is needed: (*i*) one-player games with reachability objectives and (*ii*) two-player zero-sum reachability games. In both settings, despite the Boolean nature of the objectives, a natural winning strategy consists in reaching the target as soon as possible, i.e., by minimising the length of the path from the initial vertex to the target.

One-player Reachability Games. A one-player game with a reachability objective is nothing but a connectivity problem in a graph (see [73] and [1] for surveys). If there is a path from an initial vertex to the target, there is a *simple path*, thus no memory is needed. Several algorithms have exploited this idea in order to solve the shortest-path(s) problem.[2]

As pointed in [1], the problem of finding a path from one vertex to another in a directed graph is the first problem that was identified as being complete for a natural subclass of P; it was shown to be NL-complete [49].

Two-player Zero-sum Reachability Games. Let $\mathcal{P}_1$ be the player with the reachability objective. In this two-player zero-sum framework, although it is not sufficient for $\mathcal{P}_1$ to exhibit a simple path from the initial vertex to the target, the resolution has the same flavour as the one-player case. Indeed, if $\mathcal{P}_2$, whose objective is to avoid the target, can force a loop before the target is reached, then $\mathcal{P}_2$ can avoid it forever. In particular, if $\mathcal{P}_1$ has a winning strategy, he can force a visit to the target via a simple path no matter the behaviour of $\mathcal{P}_2$. In other words, as shown in [66, 44], memoryless strategies are sufficient for $\mathcal{P}_1$ (the same holds for $\mathcal{P}_2$).

It can be shown that solving zero-sum reachability games is P-hard, even in acyclic graphs [48]. Furthermore, the alternating algorithm previously described for generalised reachability objectives runs in alternating logarithmic space when there is only one target. This implies the problem is in P, i.e., deciding the TSE problem for two-player zero-sum reachability games is P-complete.

The classical algorithm to solve two-player zero-sum reachability games is the attractor algorithm [66]. Starting from the target, it computes, layer by layer, the set of vertices from which $\mathcal{P}_1$ can win. In step j of the computation, the algorithm computes the vertices from which $\mathcal{P}_1$ can force a visit to the target in at most j steps.

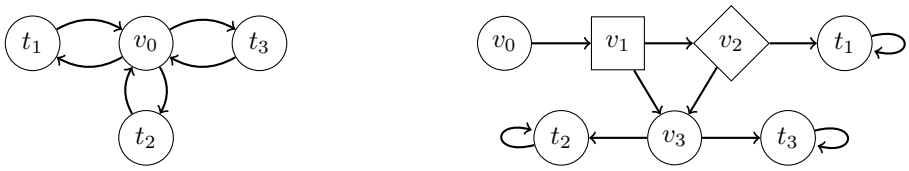
3.1.2 Sequential-memory Strategies

We now consider two situations where a memory of polynomial size is necessary: (i) one-player generalised reachability games and (ii) NEs in n-player reachability games. In both cases, the structure of the memory encodes the (fixed) order in which the targets are to be visited, and its role is to remember previously visited targets. This explains the *sequential-memory* designation: there is only one maximal chain of sets of visited targets compatible with the strategy (profile). In the case of NEs, additional memory is needed to ensure stability, via the so-called classical mechanism of *punishment*.

One-player Generalised Reachability Games. Solving a one-player game with a generalised reachability objective equates to finding a path that visits all targets in the underlying graph. The game illustrated by Figure 3a, with three targets $\mathrm{T}_j = \{t_j\}$ for $j \in [\![3]\!]$, provides a simple situation where memoryless strategies are not sufficient. Indeed, starting from v_0, any memoryless strategy visits only one target. To ensure a visit to all targets, one should first order them, for instance $(\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3)$.

In the present example, any order works. However, this is not the case in the game illustrated in Figure 1a, in which target $\mathrm{T}_3 = \{t_3\}$ must be the last to be reached. Once the order is fixed, the memory indicates the next target to be reached (via a memoryless strategy). In general, a memory of size k, encoded via a Mealy machine similar to the one in Figure 1b, is sufficient to win in a one-player game with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$.

[2] Let us cite Bellman-Ford algorithm [38, 6], Dijkstra's algorithm [31], the Floyd–Warshall algorithm [37, 72, 64] (also used to compute the transitive closure of a graph), and the closely related Kleene's algorithm used to convert a deterministic finite automaton into a regular expression [51].



(a) A one-player generalised reachability game where a sequential-memory strategy suffices to win.

(b) A three-player reachability game where punishment memory is needed to ensure NE stability.

Figure 3 Examples where memoryless strategies do not suffice.

From the discussion about SAT in Section 2, we already know that solving one-player generalised reachability games is NP-hard. In fact, the problem is NP-complete: the alternating algorithm proposed in [35, 36] to solve two-player zero-sum generalised reachability games (cf. Section 2) uses only non-determinism, not alternation, when applied to one-player games, providing the desired result.

Nash Equilibria in n-player Reachability Games. We now consider the CEE problem for NEs. Strategy profiles that solve this problem in n-player reachability games may require memory, the role of which is two-fold. The first role matches that of a sequential memory. Consider the arena illustrated by Figure 3a, interpreted as a three-player game where $\mathcal{P}_1$ controls all vertices and the targets of the players are given by $\mathrm{T}_i = \{t_i\}$ for $i \in [\![3]\!]$. Memory is necessary to obtain an NE from v_0 that visits all targets, as described in the one-player generalised reachability setting.

The second role of memory is to ensure the stability of the equilibrium. Consider the three-player reachability game on the arena of Figure 3b. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ control circle, square and diamond vertices respectively and the target of $\mathcal{P}_i$ is $\mathrm{T}_i = \{t_i\}$. An NE from v_0 such that $\mathcal{P}_1$ wins requires memory. The only play that satisfies this requirement is $v_0 v_1 v_2 t_1^\omega$. However, $\mathcal{P}_1$ cannot ensure the stability of this outcome with a memoryless strategy: from v_3, $\mathcal{P}_1$ has to remember if $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) tried to deviate to v_3 and has to choose t_3 (resp. t_2) to avoid $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) from having a profitable deviation. Let σ_1 denote the latter strategy of $\mathcal{P}_1$ and σ_2 (resp. σ_3) the memoryless strategy of $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) defined by $\sigma_2(v_1) = v_2$ (resp. $\sigma_3(v_2) = t_1$). One can check that $(\sigma_1, \sigma_2, \sigma_3)$ is an NE.

In general, outcomes of NEs can be characterised using the values of the coalition games $\mathcal{G}_i$ for each $i \in [\![n]\!]$. A play ρ is an outcome of an NE if and only if, for each player i who loses along ρ, the value of ρ_ℓ in $\mathcal{G}_i$ for $\mathcal{P}_i$ is 0 for all $\ell \in \mathbb{N}$. Any NE outcome satisfies this characterisation as otherwise a losing player would have a profitable deviation by playing his winning strategy in his coalition game $\mathcal{G}_i$ from a vertex with a value of 1 in $\mathcal{G}_i$. Conversely, the following strategy profile is an NE with outcome ρ: all players follow ρ and play an optimal strategy in $\mathcal{G}_i$ if $\mathcal{P}_i$ deviates from ρ, i.e., the coalition *punishes* $\mathcal{P}_i$. This behaviour ensures the stability of the equilibrium. This construction is inspired by the proof of the folk theorem in repeated games [39, 59]. In the NE in the game of Figure 3b defined above, from v_3, $\mathcal{P}_1$ punishes $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) if $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) deviates to v_3 by using his optimal strategy in $\mathcal{G}_2$ (resp. $\mathcal{G}_3$).

Using the characterisation above, we can show that, from any NE outcome, we can derive an NE outcome with a simple structure and the same visited targets. These outcomes are such that the path between two consecutive targets is simple and the last target is followed by a lasso hc^ω such that hc is a simple history. It follows that such outcomes have a representation of polynomial size.

From these simple outcomes, we can design a strategy profile solving the CEE problem for NEs with memory size quadratic in the number of players (and independent of the size of the arena) as follows [55]. Memory states keep track of the next target vertex and who last controlled a vertex of the (simple) path to it. If this path is exited by $\mathcal{P}_i$, then the memory detects $\mathcal{P}_i$ deviated and is updated to its $\mathcal{P}_i$ punishment state; the coalition of other players punish $\mathcal{P}_i$ using a *memoryless* punishing strategy. We remark that if $\mathcal{P}_i$ deviates without exiting the path to the next target, then it is not necessary for the other players to punish $\mathcal{P}_i$ and they continue to try progressing along the path to the next target. This *punishment memory* is not required for players who see their target, as there can be no profitable deviations for them. In particular, if all players see their target, then no punishment memory is needed and a memory size matching the number of players is sufficient as in one-player generalised reachability games. Furthermore, it is known that in every n-player reachability game there exists an NE with memory at most linear in the size of the game [19].

The CEE problem for NEs is NP-complete [14, 18]. The characterisation of NE outcomes and the existence of polynomial-size lasso outcomes above imply that the CEE problem for NEs in n-player reachability games is in NP. Hardness can be recovered from the NP-completeness of the TSE problem in one-player generalised reachability games. The player wins the one-player generalised reachability game with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_n]$ if and only if there exists an NE such that all players win in the n-player reachability game on the same arena such that $\mathcal{P}_i$, $i \in [\![n]\!]$, has T_i as target.

3.1.3 Branching-memory Strategies

We now consider two situations where a memory of exponential size is necessary: *(i)* two-player zero-sum generalised reachability games and *(ii)* SPE in n-player reachability games. Again, the memory is used to remember the set of visited targets. In the two-player setting with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$, the memory of $\mathcal{P}_1$ needs to adapt to the choices of $\mathcal{P}_2$. This leads to an exponential blow up in the memory as the whole lattice of subsets of $[\![k]\!]$ has to be remembered. This differs from the one-player setting discussed in Section 3.1.2, where only a chain of the lattice was sufficient. Regarding SPE, players may have to adapt their behaviour depending on the different subgames. These phenomena motivate the name of *branching memory*.

Two-player Zero-sum Generalised Reachability Games. Solving a two-player zero-sum game with a generalised reachability objective is more involved than one could expect [36]. From the discussion about QBF in Section 2, we know that the problem is PSPACE-complete. To exhibit a game where an exponential memory is needed for $\mathcal{P}_1$, we rely on the reduction from the QBF problem to two-player zero-sum generalised reachability games presented in Section 2 (involving the graph depicted in $\mathbf{QBF}_1$ from the small dashed rectangle of Figure 2). We consider the two-player game derived, via this reduction, from the formula:

$$\psi = \forall y_1 \forall y_2 \ldots \forall y_k \; \exists z_1 \exists z_2 \ldots \exists z_k \; \bigwedge_{i=1}^{k} (y_i \vee z_i) \wedge (\neg y_i \vee \neg z_i). \tag{1}$$

We first note that ψ holds. Thus, $\mathcal{P}_1$ (the existential player) has a winning strategy to visit all the targets. To win, $\mathcal{P}_1$ has to assign, to each variable z_i, the negation of the assignment of y_i. A *branching* memory of size 2^k is thus necessary to react to all potential assignments of $\mathcal{P}_2$. In particular, the order in which $\mathcal{P}_1$ visits the targets cannot be fixed a priori.

Notice that all the clauses of the formula ψ from (1) are of size 2. From [4], we know that such a QBF instance can be solved in linear time.[3] At first glance, this could look paradoxical with the memory requirement of $\mathcal{P}_1$ above. But as zero-sum generalised reachability games are determined, i.e., one of the two players has a winning strategy from any initial vertex, we can adopt the point of view of $\mathcal{P}_2$. From [4, Theorem 2], we can deduce that when a formula with at most two literals per clause is not satisfied, then $\mathcal{P}_2$ has a winning strategy in the corresponding game with memory of size at most 2.

In the general setting of two-player zero-sum generalised reachability games, both players may require memory of exponential size. Tight memory bounds are provided in [35, 36]. Regarding computational complexity, we note that PSPACE-hardness is only known when targets contain at least three vertices. The particular case where each target is composed of only one vertex (resp. two vertices) is solvable in polynomial time (resp. is still an open problem) [36].

Subgame Perfect Equilibria in n-player Reachability Games. Since SPEs are NEs, strategy profiles that solve the CEE problem for SPEs in n-player reachability games are no less complex than those needed for NEs. In particular, punishment memory may be necessary. Furthermore, due to the fact that a player should act rationally in all subgames, *branching memory* is needed to choose the next action, and may be more sophisticated than for NEs. It may be exponential for reasons similar to two-player zero-sum generalised reachability games.

In this setting, the CEE problem is PSPACE-complete [16]. The PSPACE membership relies on the following property: if there exists an SPE that solves the problem, then there exists an SPE that solves the problem and has a finite representation. This representation is a set of lassoes of polynomial length, mainly one per element of $[\![n]\!] \times V \times 2^n$, that satisfy some property on the players' gains. This characterisation allows the design of an alternating algorithm that runs in polynomial time. More details can be found in [16].

PSPACE-hardness is due to a reduction from the QBF problem (explained in Section 2). From a quantified formula $\psi = Q_1x_1\ Q_2x_2\ \ldots\ Q_mx_m\ C_1 \wedge\ C_2 \wedge \ldots \wedge C_k$ we build a $(k+2)$-player reachability game on the graph illustrated by the big dashed rectangle ($\mathbf{QBF_2}$) of Figure 2. In addition to the structure of $\mathbf{QBF_1}$, we consider one additional vertex per clause and two more vertices $t_\forall$ and $t_\exists$, linked by the edges depicted in the figure. This is a $(k+2)$-player reachability game with initial vertex Q_1 such that $\mathcal{P}_i$, $i \in [\![k]\!]$ represents the clause C_i, controls the vertex C_i and wants to visit a literal of his clause or vertex $t_\forall$. The other two players are $\mathcal{P}_\exists$ and $\mathcal{P}_\forall$: $\mathcal{P}_\exists$ (resp. $\mathcal{P}_\forall$) controls existential (resp. universal) vertices and wants to reach $t_\exists$ (resp. $t_\forall$). In an equilibrium, $\mathcal{P}_\exists$ cannot win if some clause C_i is not satisfied since $\mathcal{P}_i$ should move to $t_\forall$ from vertex C_i. Moreover, $\mathcal{P}_\exists$ wins if and only if $\mathcal{P}_\forall$ loses. With this construction, it can be shown that ψ is true if and only if there exists an SPE such that all players win except $\mathcal{P}_\forall$. Notice that this reduction is no longer correct if we replace SPE by NE: the subgame rationality imposed by SPEs plays a crucial role. Indeed, let us consider the formula $\psi = \forall x_1 \exists x_2 (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$ and the memoryless strategy profile σ such that $\sigma_\forall(\forall_1) = \neg x_1$, $\sigma_\exists(\exists_2) = x_2$, $\sigma_1(C_1) = C_2$, $\sigma_2(C_2) = C_3$ and $\sigma_3(C_3) = t_\exists$. We can prove that σ is an NE such that all players but $\mathcal{P}_\forall$ win, although ψ is false. In particular, $\mathcal{P}_\forall$ has no incentive to move towards x_1 since, even if $\mathcal{P}_2$ does not visit his target along this outcome, he still chooses to move towards $t_\exists$. However,

[3] The algorithm is based on a smart analysis of the strongly connected components of a directed graph associated with the formula.

this latter behaviour is not rational in the subgames induced by the $\mathcal{P}_\forall$'s choice to go to x_1, since $\mathcal{P}_2$ should choose $t_\forall$ and win. Thus σ is not an SPE. Finally, remark that deciding if there exists an SPE such that all players win is easier; it amounts to deciding if there exists a winning strategy for the player of the one-player generalised reachability game with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_n]$. That can be done with a nondeterministic polynomial time algorithm.

Thanks to the finite representation of an SPE mentioned before, it can be deduced from [16] that strategy profiles with memory size exponential in the number of players and polynomial in the size of the arena are sufficient to solve the CEE problem for SPEs.

3.2 Repeated Reachability

We now consider the (generalised) Büchi objective functions. Let $\mathrm{T} \subseteq V$ be a target, the *Büchi objective (for* T*)* consists in visiting T infinitely often. Formally we have that $\mathsf{Büchi}[\mathrm{T}](\rho) = 1$, if for all $j \in \mathbb{N}$ there exists a position $\rho_\ell \in \mathrm{T}$ with $\ell > j$, otherwise $\mathsf{Büchi}[\mathrm{T}](\rho) = 0$. Let $\mathrm{T}_1, \ldots \mathrm{T}_k \subseteq V$ be target sets. The *generalised Büchi objective (for* $\mathrm{T}_1, \ldots \mathrm{T}_k$*)* consists in visiting all the sets T_j infinitely often. Formally we have that $\mathsf{GBüchi}[\mathrm{T}_1, \ldots, \mathrm{T}_k](\rho) = \prod_{j=1}^{k} \mathsf{Büchi}[\mathrm{T}_j](\rho)$.

This section follows a line of progression similar to that of Section 3.1.

3.2.1 Memoryless Strategies

As in Section 3.1.1, memoryless strategies are sufficient in two situations: (*i*) one-player games with Büchi objectives and (*ii*) two-player zero-sum Büchi games. These memoryless strategies are essentially reachability strategies for targets that can be visited infinitely often.

3.2.2 Looping-sequential-memory Strategies

Let us now consider (*i*) one-player generalised Büchi games and (*ii*) NEs in n-player Büchi games, the counterparts of the games studied in Section 3.1.2, and (*iii*) two-player generalised Büchi games, the counterpart of the games studied in the first part of Section 3.1.3. For (*i*) and (*ii*), the required memory is similar to the reachability case. Indeed, in the two examples of Figure 3 the (generalised) reachability objectives can be translated into (generalised) Büchi objectives without affecting the memory requirements of the players. Although Büchi objectives appear more complex than reachability objectives, in both cases, the decision problems are simpler and can be solved in polynomial time, whereas the reachability variants are NP-complete. For (*iii*), the decision problem can also be solved in polynomial time, with a memory of polynomial size, even though the generalised reachability variant is PSPACE-hard and requires strategies with exponential memory.

One-player Generalised Büchi Games. Solving a one-player game with a generalised Büchi objective $\mathsf{GBüchi}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$ amounts to finding a reachable strongly connected component (SCC) which contains elements of each target $\mathrm{T}_1, \ldots, \mathrm{T}_k$. This can be done in polynomial time [65]. The structure of the memory encodes a (fixed) order in which the targets have to be repeatedly visited. We speak here of *looping sequential memory* as the reachability scheme has to be repeated. As an example, let us consider the one-player game depicted on Figure 3a with objective $\mathsf{GBüchi}[\{t_1\}, \{t_2\}, \{t_3\}]$. The memory structure of a winning strategy can be derived from the Mealy machine of Figure 1b with an additional transition from m_2 to m_0 when reading edge (v_0, t_3) to complete the loop.

An alternative polynomial time algorithm to solve a one-player generalised Büchi game is to transform it into a one-player (simple) Büchi game of polynomial size and then solve this Büchi game. Given a game with objective $\mathsf{GBüchi}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$, the Büchi game is obtained by

juxtaposing k copies of the original game. We move from the jth copy to the $(j+1)$th copy (when $j \neq k$) or to the first (when $j = k$) whenever T_j is visited within the jth copy. The (simple) Büchi condition on this expanded game consists in visiting infinitely often the first copy of T_1. Details of the construction and its correctness can be found in [5, Theorem 4.56].

Such an elegant *polynomial-size* construction would not work for a generalised reachability objective. This construction relies on the fact that if the objective $\mathsf{GBüchi}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$ is satisfied, we can visit targets in any order as there are vertices of each target that are connected. In particular, it is suitable to first visit T_1, then T_2 and so on. By contrast, recall that enforcing $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$ may require visiting targets in a specific order (see Example 3). The analogue of the construction above for generalised reachability games is to include the sets of previously visited targets in the vertices of the arena, which comes at an exponential price [35, 36].

Equilibria in n-player Büchi Games. Let us now turn to NEs in n-player Büchi games. In this setting, a polynomial-size memory is sufficient [70]. Once more, SCCs play a crucial role to obtain a polynomial algorithm. A deterministic polynomial algorithm is presented in [70]. This algorithm relies on a characterisation of outcomes of NEs as explained for n-player reachability games and on an algorithm tailored for the Büchi objective that finds an adequate reachable SCC in refined arenas of the game.

In contrast to the NE case, memory requirements as well as the exact complexity class of the CEE problem for SPEs in n-player Büchi games are still open. Nevertheless, this latter problem is in NP since it is the case for the more general parity games [15].

Two-player Generalised Büchi Games. Two-player zero-sum generalised Büchi games can also be solved in polynomial time and a memory of polynomial size is sufficient [32, 23]. The construction discussed above for one-player generalised Büchi games can be adapted to the two-player zero-sum setting. As memoryless strategies suffice to win in (simple) Büchi games, it follows from the construction that a polynomial-size memory suffices to win in generalised Büchi games. The role of the memory is to ensure that T_1 is visited, then T_2 until T_k and then restart from T_1.

3.3 Reaching Targets Quickly: Shortest-path Games

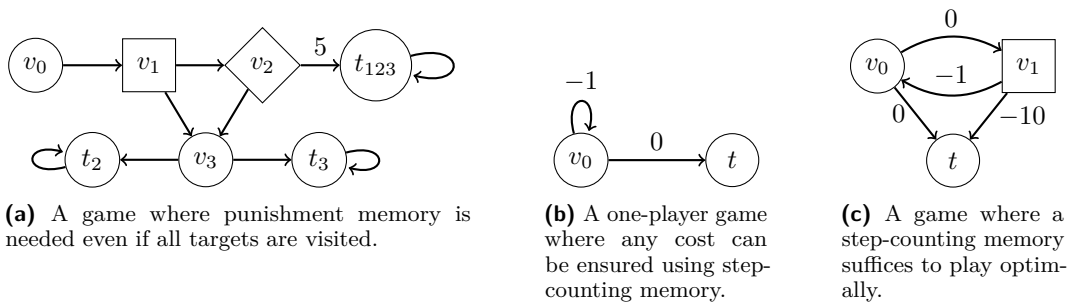
In this section, we focus on reaching a target as soon as possible. In order to formalise this idea, we first need to enrich our arenas with weights on edges. A *weighted arena* is couple $(\mathcal{A}, w)$ where $\mathcal{A}$ is an arena and $w \colon E \to \mathbb{Z}$ is a weight function.

Let $\mathrm{T} \subseteq V$ be a target. We introduce the *shortest-path objective* described by the cost function $\mathsf{SPath}_w[\mathrm{T}]$ that players intend to minimise. It is a quantitative variant of the Boolean reachability objective function. Formally, $\mathsf{SPath}_w[\mathrm{T}] : \mathrm{Plays} \to \overline{\mathbb{Z}}$ is defined by:

$$\mathsf{SPath}_w[\mathrm{T}](\rho) = \begin{cases} \sum_{j=0}^{\ell-1} w(\rho_j, \rho_{j+1}) & \text{if } \ell \text{ is the least index such that } \rho_\ell \in \mathrm{T} \\ \infty & \text{otherwise.} \end{cases}$$

Definitions using gain functions can be recovered for cost functions by reversing inequalities (e.g., solution concepts and the CEE problem).

In Section 3.3.1, we study shortest-path games whose weight function is restricted to non-negative weights, while Section 3.3.2 considers unrestricted weights.



(a) A game where punishment memory is needed even if all targets are visited.

(b) A one-player game where any cost can be ensured using step-counting memory.

(c) A game where a step-counting memory suffices to play optimally.

Figure 4 Weighted arenas for shortest-path games. Unlabelled edges have a weight of 1.

3.3.1 Same Strategies as the Boolean Case: Non-negative Weights

First, we consider the case where $w\colon E \to \mathbb{N}$. In a one-player or two-player zero-sum setting, there is always a memoryless optimal strategy in shortest-path games. The memory needed to solve the natural shortest-path variants of the CEE problem differs only slightly from the Boolean case. Namely, for NEs, punishment memory could be required even when all players visit their target.

Let us consider the three-player shortest-path game played on the arena of Figure 4a with initial vertex v_0. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ control circle, square and diamond vertices respectively and their targets are $\{t_{123}\}$, $\{t_2, t_{123}\}$ and $\{t_3, t_{123}\}$. To obtain an NE visiting targets of all players, $\mathcal{P}_1$ needs memory. Indeed, the only possible such outcome is $v_0 v_1 v_2 t_{123}^{\omega}$ with a cost of 7. However, if $\mathcal{P}_1$ uses a memoryless strategy and chooses t_2 (resp. t_3) from v_3 then, $\mathcal{P}_3$ (resp. $\mathcal{P}_2$) should deviate towards v_3 to obtain a cost of 3. Punishment memory, matching that required in the game illustrated in Figure 3b, is needed.

One could say that nothing differs in terms of computational complexity for the TSE problem in shortest-path one-player or two-player zero-sum games (e.g., [31, 52, 50, 21]). In general, the philosophy of the algorithms remains the same, as the algorithms for the Boolean case already compute shortest paths in some sense.

Let us now turn to the shortest-path variants of the CEE problem. In this case, nothing differs in terms of complexity classes. The NP membership for NEs in n-player shortest-path games can be derived from the arguments used in the Boolean case. As the hardness argument in the Boolean setting of one-player generalised reachability games is not applicable in this case, NP-hardness relies on a reduction from the SAT problem. This reduction is in the same spirit as the construction presented for SPEs but restricted to existential formulas [18]. To obtain PSPACE-completeness for SPEs in n-player shortest-path games, non-trivial adaptations of the Boolean case are needed (specifically for PSPACE membership), see [17].[4]

We now mention another setting where an exponential memory size is needed. In the Boolean setting, the simplest such case is when solving two-player zero-sum generalised reachability games. We present here a multi-dimensional variant of the TSE problem where an exponential size memory is needed even with a single target T.

We consider a two-player arena equipped with a multi-dimensional weight function $w\colon E \to \mathbb{N}^k$. The (multi-dimensional) cost function of $\mathcal{P}_1$ is such that, for all plays $\rho \in \mathrm{Plays}$, $\mathsf{SPath}_w[\mathrm{T}](\rho) = (\mathsf{SPath}_{w_1}[\mathrm{T}](\rho), \ldots, (\mathsf{SPath}_{w_k}[\mathrm{T}](\rho))$. Given $\alpha \in \mathbb{N}^k$, we study the existence of a strategy σ_1 of $\mathcal{P}_1$ such that for all σ_2 of $\mathcal{P}_2$, $\mathsf{SPath}_{w_j}[\mathrm{T}](\langle \sigma_1, \sigma_2 \rangle) \leq \alpha_j$ for each $j \in [\![k]\!]$.

[4] Notice that in [17], authors assume that $w(e) = 1$ for all $e \in E$.

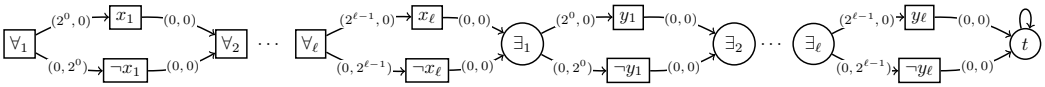


Figure 5 A multi-dimensional shortest-path game where an exponential memory size is required.

Consider the game played on the arena depicted in Figure 5 where $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) controls circles (resp. squares) and $\mathrm{T} = \{t\}$. This illustrates a situation where exponential memory is needed: a strategy that solves the game for the threshold $(2^\ell - 1, 2^\ell - 1)$ requires a memory with size exponential in the number of vertices. Intuitively, $\mathcal{P}_1$ has to remember the choice of $\mathcal{P}_2$ in vertex $\forall_j$ and do the opposite in $\exists_j$ for all $j \in [\![\ell]\!]$. This multi-dimensional version of the TSE problem is PSPACE-complete [22]. The PSPACE-hardness, proved via a reduction from the *quantified subset-sum problem* [67, Lemma 4], holds even if $|\,\mathrm{T}\,| = 1$ and $k = 2$.

3.3.2 Step-counting-memory Strategies: Unrestricted Weights

We now turn to the case where $w\colon E \to \mathbb{Z}$. In this setting, complex phenomena already occur in one-player shortest-path games. We consider the one-player shortest-path game on the arena depicted in Figure 4b with target $\{t\}$. There is no optimal strategy to reach t from v in this game. Indeed, given any strategy σ_1 ensuring the cost α, there exists a *better* strategy σ_1' ensuring $\alpha - 1$ (remember that we aim at minimising the cost). For a given threshold α, there exists a strategy ensuring α, but this strategy needs memory. The memory is used to count the number of loops to take before moving to t.

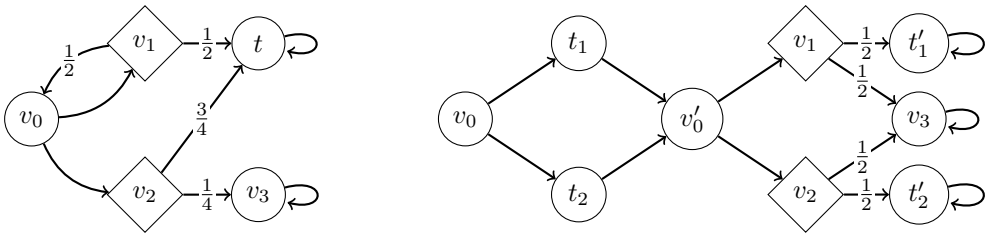
We call strategies that follow a memoryless strategy that accumulates negative weight for a fixed number of steps before switching to a memoryless shortest-path strategy a strategy with *step-counting memory*. If there exists a strategy ensuring a threshold α (encoded in binary), then there is a strategy with step-counting memory that ensures α with a memory of pseudo-polynomial size, i.e., of size exponential in the size of the encoding of α (recall that the size of the memory refers to the size of the Mealy machine).

When considering two-player zero-sum shortest-path games, similar, but slightly more subtle phenomena occur. Let us consider the shortest-path game on the arena of Figure 4c where $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) controls the circle (resp. square) vertices with target $\{t\}$. Starting from v_0, $\mathcal{P}_1$ has an optimal strategy σ_1 ensuring -10. At first glance, one could think that the memoryless strategy consisting in reaching v_1 is optimal, but in this case, t will never be reached as $\mathcal{P}_2$ will always go back to v_0. An optimal strategy σ_1 consists in counting 10 visits to v_1 before reaching t. Facing σ_1, the optimal (memoryless) strategy for $\mathcal{P}_2$ is to directly reach t, whose resulting outcome is a simple path. Two-player zero-sum shortest-path games can be solved in pseudo-polynomial time. In general, $\mathcal{P}_1$ does not necessarily have an optimal strategy, but if he has one, there is one with step-counting memory of pseudo-polynomial size. This contrasts with $\mathcal{P}_2$ who always has an optimal memoryless strategy [20, 21].

On the one hand, the multi-dimensional variant of the TSE problem discussed in Section 3.3.1 is undecidable if we allow unrestricted weights [61]. On the other hand, regarding NEs in n-player shortest-path games, the example from Figure 4b directly implies that NEs are not guaranteed to exist in this setting.

4 Stochastic Arenas

In this section, we consider arenas with randomised transitions. We focus on the case where one player progresses in a stochastic environment, i.e., one-player stochastic games, also known as Markov decision processes. In Section 4.1, we introduce our formal model as an



(a) An MDP where $\mathcal{P}_1$ has an incentive to move to v_2 only if the number of steps to visit t is limited.

(b) An MDP where both randomisation and memory are necessary to visit $\{t_1, t'_1\}$ and $\{t_2, t'_2\}$ with probability at least $\frac{3}{4}$.

Figure 6 Example MDPs. Circles and diamonds resp. represent $\mathcal{P}_1$ and stochastic vertices.

extension of the model of Section 2 and describe the problems we consider. Section 4.2 is the counterpart of Section 3 for the problems described in this section; we overview the complexity of strategies necessary to address these problems and comment on algorithms.

4.1 Markov Decision Processes

In this section, we extend notions of Section 2 to the one-player probabilistic setting.

Probability Distributions. Let A be a finite or countable set. We let $\mathcal{D}(A)$ denote the set of (discrete) probability distributions over A, i.e., functions $p\colon A \to [0,1]$ with $\sum_{a\in A} p(a) = 1$. For any $p \in \mathcal{D}(A)$, we let $\mathsf{supp}(p) = \{a \in A \mid p(a) > 0\}$ denote the support of p.

Markov Decision Processes. Let (V, E) be a directed graph as in Section 2. A *Markov decision process (MDP)* or one-player stochastic arena (on (V, E)) is a tuple $\mathcal{M} = (V_1, V_\Delta, E, \Delta)$ where (V_1, V_Δ) is a partition of V into vertices of $\mathcal{P}_1$ and stochastic vertices and $\Delta\colon V_\Delta \to \mathcal{D}(V)$ is a stochastic transition function such that $\mathsf{supp}(\Delta(v)) = \{v' \in V \mid (v, v') \in E\}$ for all $v \in V_\Delta$ (i.e., $\Delta(v)$ assigns a positive probability to all successors of v and only to them). We assume that MDPs are deadlock-free. MDPs subsume one-player game arenas.

▶ Remark 5. MDPs can alternatively be defined by a set of states, a set of actions, and a probabilistic transition function that assigns distributions over successor states to pairs of states and actions. The definition of MDPs presented here is equivalent [27], in addition to being compatible with the definitions of the previous section.

Definitions of plays and histories are identical to those provided in Section 2 for non-stochastic arenas. We will use the same notation. A *multi-weighted MDP* is a pair $(\mathcal{M}, w)$ where $w\colon E \to \mathbb{N}^k$ is a multi-dimensional (non-negative) weight function. An *event* is a set of measurable plays (for the Borel σ-algebra on the set of plays).

We illustrate two MDPs in Figure 6. Plays in MDPs proceed differently than in deterministic games: due to the presence of stochastic vertices, there may be several outcomes from an initial vertex for a given strategy of $\mathcal{P}_1$. At each round of a play, if the current vertex v is controlled by $\mathcal{P}_1$, then he chooses the next vertex, however if v is stochastic, then the next vertex is chosen randomly according to the distribution $\Delta(v)$. For instance, in the MDP of Figure 6a, the next vertex from v_1 is chosen uniformly at random between v_0 or t.

Randomised Strategies. In MDPs, we consider more general strategies than in Section 3. Randomised strategies yield a distribution over the possible choices of $\mathcal{P}_1$ instead of providing an action deterministically. These strategies allow a trade-off between different (and possibly

incompatible) goals. For instance, in the MDP depicted in Figure 6b, if $\mathcal{P}_1$ wants to ensure that the states t_1 and t_2 are each visited with probability $\frac{1}{2}$, he cannot do so without randomisation. In this example, both states t_1 and t_2 cannot be visited along the same path.

Formally, a (randomised) strategy (of $\mathcal{P}_1$) is a function $\sigma\colon \mathrm{Hist}_1 \to \mathcal{D}(V)$ such that, for all histories $h = h_0 \dots h_\ell \in \mathrm{Hist}_1$, $\mathsf{supp}(\sigma(h)) \subseteq \{v \in V \mid (h_\ell, v) \in E\}$. A strategy σ is *pure* if it does not use randomisation, i.e., if for all histories $h \in \mathrm{Hist}_1$, $|\mathsf{supp}(\sigma(h))| = 1$. Memoryless (randomised) strategies are defined in the same way as previously.

We say that a randomised strategy is *finite-memory* if it can be encoded by a (stochastic) Mealy machine $\mathfrak{M} = (\mathrm{M}, m_0, \mathsf{up}, \mathsf{nxt})$ where M, m_0 and up are defined in the same way as previously and $\mathsf{nxt}\colon \mathrm{M} \times V_1 \to \mathcal{D}(V)$ is changed to output distributions over successors vertices rather than a single successor. The strategy induced by a Mealy machine is defined similarly to the case of deterministic arenas, except it outputs distributions instead of vertices.

We remark that finite-memory randomised strategies can be defined in alternative ways, e.g., by also including randomisation in the initialisation or in the updates of the Mealy machine. Such alternatives are not equivalent to the one given above [56]. We do not consider richer models: the one above is sufficient to solve the problems considered in this section.

Outcomes. Let σ be a strategy of $\mathcal{P}_1$ and v_0 be an initial vertex. By fixing a strategy and initial vertex, we obtain a purely stochastic process, i.e., a Markov chain over the set of histories of the MDP. Intuitively, for each history h, one can assign a probability to the plays that extend h, defined as the product of the probability each edge along the history being taken according to the strategy of $\mathcal{P}_1$ and the stochastic transition function. This distribution extends in a unique manner to all events. We denote it by $\mathbb{P}^{\sigma}_{v_0}$. We refer the reader to, e.g., [5], for a formal definition of this distribution.

We stress that the nature of outcomes differs from the non-stochastic case. On the one hand, the outcome of a strategy profile from a vertex is a singular play in the non-stochastic setting. On the other hand, with MDPs, we obtain a Markov chain after fixing the strategy of $\mathcal{P}_1$ and the initial vertex. Furthermore, there can be infinitely many plays resulting from the Markov chain, even when the strategy of $\mathcal{P}_1$ is pure. This can be observed on the MDP of Figure 6a by considering the initial vertex v_0 and the pure memoryless strategy of $\mathcal{P}_1$ that moves from v_0 to v_1. Due to the stochastic vertex v_1, there are infinitely many plays played according to this strategy.

Relevant Events and Problems. In the following, we consider two types of events derived from gain and cost functions. Let T denote a target set and w a (one-dimensional) weight function. The first type of event relates to the purely Boolean reachability objective. Such an event is a set $\{\rho \in \mathrm{Plays} \mid \mathsf{Reach}[\mathrm{T}](\rho) = 1\}$. For conciseness, we will abusively denote this event by $\mathsf{Reach}[\mathrm{T}]$. The second type of event pertains to shortest-path problems, and these events are given by sets $\{\rho \in \mathrm{Plays} \mid \mathsf{SPath}_w[\mathrm{T}](\rho) \leq \alpha\}$ for some threshold $\alpha \in \mathbb{N}$.

We study both one-dimensional and multi-dimensional problems. In the one-dimensional case, we are interested in the maximal probability of an event and computing strategies that realise this probability. A strategy is *optimal* if it attains this probability.

In the multi-dimensional case, we want to satisfy lower-bound constraints on the probability of several events. More precisely, let $v_0 \in V$ be an initial vertex, $\Omega_1, \dots, \Omega_k$ be events (of the same type) and $\beta_1, \dots, \beta_k \in [0, 1]$ be thresholds. We want to decide the existence of strategy σ such that for all $j \in [\![k]\!]$, $\mathbb{P}^{\sigma}_{v_0}(\Omega_j) \geq \beta_j$, and synthesise such a strategy if it exists.

We remark that there also exist metrics other than probability to measure the quality of strategies that we do not discuss here. For instance, given a cost function, one could aim to find a strategy minimising the expected cost (e.g., see [8] for the shortest path cost function).

4.2 Reachability Problems in Markov Decision Processes

In this section, we overview the problems introduced at the end of the previous section. In Section 4.2.1, we present one-dimensional problems for which pure strategies suffice. We follow with their multi-dimensional counterparts in Section 4.2.2, for which randomised strategies are required. In both sections, we first discuss the Boolean reachability objective, then the shortest-path variant. For the former, memoryless strategies suffice in the one-dimensional case, and the same memory structure as in two-player generalised reachability games suffices in the multi-dimensional case. In the shortest-path case, the memory keeps track of the accumulated weight (for each dimension) in addition to the information used in the unweighted case. We fix an MDP $\mathcal{M} = (V_1, V_\Delta, E, \Delta)$ for this section.

4.2.1 Pure Strategies: One-dimensional Problems

Memoryless Strategies: Reachability. The maximum reachability probability can always be achieved using a *pure memoryless* strategy. Such a strategy can be obtained once the optimal probabilities have been computed for all vertices. First, we restrict edges that leave $\mathcal{P}_1$ vertices to those that enter a successor vertex with highest maximum probability. A memoryless strategy is optimal if and only if it uses only the restricted edges and, from all vertices with a positive maximum reachability probability, there is a path according to the strategy to a target state. A natural way to compute such a strategy is to select one such that, from all states with positive maximum reachability probability, there is a path to a target that is as short as possible (e.g., by a backwards induction approach).

Due to the nature of outcomes in this setting, in contrast with the non-stochastic setting, it is not possible to enforce that targets are reached by simple paths, even if a target is reached almost-surely. This can be observed on the MDP of Figure 6a. The (only) optimal memoryless strategy of $\mathcal{P}_1$ is the one that moves from v_0 to v_1, and it ensures t is reached with probability 1 from v_0 (and v_1). As explained previously, there are infinitely many plays played according to this strategy, and the time to reach a target in these plays is not bounded from above.

Computing the maximum probability of reaching a target set T in $\mathcal{M}$ can be done in polynomial time via linear programming [5]. Furthermore, deciding whether there is a strategy ensuring that a target is reached almost-surely is P-hard (for reasons similar to two-player zero-sum Boolean reachability games). The existence of strategies that ensure that T is reached almost-surely depends only on the graph underlying $\mathcal{M}$ and the partition of vertices, and is independent of the probabilities assigned by Δ. We remark that this special case is no easier than the general one from a complexity standpoint.

Counting-memory Strategies: Weight-constrained Reachability. Let T be a target set, $\alpha \in \mathbb{N}$, $\beta \in [0, 1]$ be thresholds and w be a one-dimensional weight function. In general, strategies that maximise the probability of the event $\Omega = \{\rho \in \text{Plays} \mid \mathsf{SPath}_w[\mathrm{T}](\rho) \leq \alpha\}$ require memory.

To illustrate this, let us consider the MDP of Figure 6a, assume that all edges have a weight of 1 (i.e., we count steps), and that $\mathcal{P}_1$ wants to maximise the probability of reaching t within four steps from v_0. We only describe the strategy in v_0 when two or four steps remain, as it is the only decision state and two steps are required to return to it. If two steps remain, the only optimal move is to go from v_0 to v_2, as it ensures the target is reached with probability $\frac{3}{4}$ instead of $\frac{1}{2}$. If four steps remain, moving to v_1 is the only optimal move, as it ensures a greater probability of success than $\frac{3}{4}$ due to the probability of success from v_0

when two steps remain being $\frac{3}{4}$. We observe that optimal strategies in this setting may select edges that would otherwise not be chosen when maximising the probability of $\mathsf{Reach}[\{t\}]$. In other words, $\mathcal{P}_1$ has an incentive to select riskier moves due to the weight constraints.

Deciding whether the maximum probability of Ω is at least β is PSPACE-hard, even for acyclic MDPs, for which it is PSPACE-complete [45]. In full generality, the problem can be solved in time polynomial in the size of the MDP and the threshold α, i.e., in pseudo-polynomial time. To compute the maximum probability and derive a maximising strategy, we can augment vertices of the MDP with the accumulated weight from the initial vertex and make vertices with an accumulated weight exceeding α absorbing (i.e., all edges leaving these vertices return to it). This ensures the expanded MDP is finite. On this expanded MDP, the optimal probabilities and an optimal strategy can be computed using the techniques described above, by considering as targets the expanded original target vertices with an accumulated weight of at most α.

Due to memoryless strategies being sufficient to maximise reachability probabilities, it follows that finite-memory strategies of pseudo-polynomial size that keep track of the accumulated weight suffice to maximise the probability of Ω, i.e., a *counting memory* is sufficient. A memory of pseudo-polynomial size is necessary in certain cases. Whether a move is optimal in a vertex in this setting is an intrinsic property of the difference between the bound α and the weight accumulated when entering the vertex. Unlike strategies with step-counting memory presented in Section 3.3.2, the strategies here do not only switch once from one memoryless strategy to another based on the number of steps taken, but instead make decisions based on the accumulated weight. In particular, in a given state, there may be more than two different memoryless strategies assigned to the memory states of a Mealy machine encoding an optimal strategy (e.g., this would occur if there were an additional state in the MDP of Figure 6a that reaches t with probability $\frac{7}{8}$ in exactly three steps).

4.2.2 Randomised Strategies: Multi-dimensional Problems

Branching-memory Strategies: Multi-objective Reachability. We now consider multiple targets $\mathrm{T}_1, \ldots \mathrm{T}_k$ and probability thresholds $\beta_1, \ldots, \beta_k \in [0, 1]$, and the problem of existence and synthesis of a strategy σ such that for all $j \in [\![k]\!]$, $\mathbb{P}^{\sigma}_{v_0}(\mathsf{Reach}[\mathrm{T}_j]) \geq \beta_j$. Instead of maximising the probability of having $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k] = 1$, we treat each target separately. This allows one to study the trade-off between the different reachability objectives. The special case where, for all $j \in [\![k]\!]$, $\beta_j = 1$, is nonetheless equivalent to deciding the existence of a strategy ensuring that $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k] = 1$ almost-surely.

Unlike all previously introduced problems, randomised strategies are necessary to satisfy these multi-dimensional constraints, as was explained when introducing randomised strategies. Furthermore, memory may be necessary. Consider the MDP of Figure 6b and the target sets $\mathrm{T}_1 = \{t_1, t'_1\}$ and $\mathrm{T}_2 = \{t_2, t'_2\}$. To visit both target sets with probability $\frac{3}{4}$ from v_0, pure strategies do not suffice; at best, one target can be visited almost-surely and the other with a probability of $\frac{1}{2}$. Furthermore, randomised memoryless strategies can also be argued to be insufficient. A suitable strategy is one that selects a successor of v_0 uniformly at random, and then moves (deterministically) from v'_0 to v_{3-j} if t_j was visited (i.e., the memory keeps track of previously visited target sets).

Furthermore, a memory of size exponential in the number of targets is necessary and sufficient. Consider the graph $\mathbf{QBF}_1$ of Figure 2 for the formula in Equation (1), and let existential (resp. universal) vertices be controlled by $\mathcal{P}_1$ (resp. be stochastic). For the targets we considered for generalised reachability, exponential memory is necessary to ensure all targets are visited almost-surely. In general, the same type of branching memory as in two-player zero-sum generalised reachability games is sufficient.

In a nutshell, the role of the randomisation in strategies is to balance the different objectives, whereas the role of the memory in strategies is to recall the visited targets. In the special case where all targets are absorbing, no memory is required [33], therefore randomised memoryless strategies suffice.

The problem can be shown to be PSPACE-hard using the previously sketched construction for general QBF formulas [62]. The case where $\beta_1 = \ldots = \beta_k = 1$ is PSPACE-complete [62]. The case where targets are absorbing can be solved in polynomial time by linear programming. The general problem can be solved in time polynomial in the size of the MDP and exponential in the number of targets, by reduction to the case where targets are absorbing [33].

Branching-counting-memory Strategies: Multi-objective Weight-constrained Reachability. In addition to the targets and probability thresholds fixed above, we consider a multi-dimensional weight function $w\colon E \to \mathbb{N}^k$ and thresholds $\alpha_1, \ldots, \alpha_k \in \mathbb{N}$. A strategy σ such that $\mathbb{P}^{\sigma}_{v_0}(\mathsf{SPath}_{w_j}[\mathrm{T}_j] \leq \alpha_i) \geq \beta_j$ holds for all $j \in [\![k]\!]$, if it exists, can always be found with a combination of the traits of the counting-memory and branching-memory strategies from the two previous problems. In other words, for such constraints, randomised finite-memory strategies that keep track of the accumulated weight and the sets of visited targets suffice. Therefore, memory exponential in the number of targets and the encoding of the bounds α_j is sufficient. The need for an exponential-sized memory in general follows from the need for an exponential-size memory for unweighted multi-objective reachability.

These strategies can be synthesised by merging the algorithmic ideas for the two previous problems [62]. First, one augments the MDP vertices with vectors of accumulated weights for dimensions j for which the bound α_j has not been exceeded yet. Next, by defining targets for each dimension in the augmented MDP as in the one-dimensional shortest-path problem, checking the existence of a strategy and synthesising it can be done via algorithms for multi-dimensional reachability. It follows that this problem can be solved in time polynomial in the size of the MDP and exponential in the number of targets and encoding of the bounds α_j. It also inherits PSPACE-hardness from the previous problem. This problem is undecidable if we allow weights in $\mathbb{Z}$ [62].

5 Conclusion

The goal of this paper was to look at reachability-related problems through the lens of memory. Now that our journey has reached its end, we briefly comment on what lies beyond the intended scope of this paper.

We have only considered *perfect information turn-based* games. Relaxing these hypotheses induces a need for more complex strategies, even in zero-sum Boolean reachability games. For instance, in concurrent games, neither player may have a winning strategy. Furthermore, randomisation may be necessary to maximise one's probability of winning. There may not be a strategy that ensures the maximal probability of $\mathcal{P}_1$, although this probability can be approached by memoryless randomised strategies [29, 47]. Moreover, without perfect information, $\mathcal{P}_1$ may need a memory of exponential size, in addition to randomisation, to maximise the reachability probability [7]. Deciding whether $\mathcal{P}_1$ has an almost-surely winning strategy in a concurrent reachability game of imperfect information is 2-EXPTIME-complete [7], a stark contrast with the turn-based perfect information setting.

Throughout this paper, we have only dealt with *finite* arenas. Reachability problems on infinite arenas with a finite description have also garnered interest. Notable such problems include the halting problems for Turing [68] or Minsky [57] machines, which are undecidable.

The problem of solving zero-sum reachability games on the configuration graph of a pushdown automaton is EXPTIME-complete [71]. The reachability problem in Petri nets and vector addition systems was recently shown to be Ackermann-complete [54, 28]. Regarding questions related to strategy complexity in infinite games, one can mention [10] where the memory requirements of winning strategies are studied in zero-sum games where the goal is to construct (i.e., reach) a word from a regular language.

When considering strategies, we only introduced *randomisation* in situations were it was required. In particular, we did not discuss the potential trade-off between the size of a deterministic Mealy machine and the use of randomisation. In some cases, as discussed in [58, 24, 46, 25], considering randomised strategies in situations where pure strategies suffice can yield smaller memory bounds for optimal (or close-to-optimal) strategies. Specifically studying the existence of pure strategies to solve certain problems can sometimes turn out more complex than searching for randomised strategies [30].

We have highlighted some similarities in the memory structure needed to solve some problems that are often treated as different. We believe that the memory structure of strategies should be investigated deeper, in line with the effort to understand when complex strategies are required that is described in Section 1. There exist works that study alternatives to Mealy machine to represent strategies, e.g., models based on Turing machine [41], decision trees for memoryless strategies (e.g., [2, 3]) or compact counter-based models in a setting with limited resources [9].

In Section 3.2, we noted that although Büchi objectives appear more complex than reachability objectives, solving games with objectives derived from the former is often simpler than dealing with games based on the latter. A key ingredient enabling efficient algorithms to solve games involving Büchi objectives is the ability to heavily rely on SCCs. From this, one could question whether there is a relation between the SCC decomposition of a reachability game and the memory structure needed to solve it.

References

1. Eric Allender. Reachability problems: An update. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *Computation and Logic in the Real World, Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18-23, 2007, Proceedings*, volume 4497 of *Lecture Notes in Computer Science*, pages 25–27. Springer, 2007. doi:10.1007/978-3-540-73001-9_3.
2. Pranav Ashok, Mathias Jackermeier, Jan Kretínský, Christoph Weinhuber, Maximilian Weininger, and Mayank Yadav. dtControl 2.0: Explainable strategy representation via decision tree learning steered by experts. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Proceedings (Part II) of the 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2021, Held as Part of ETAPS 2021, Luxemburg City, Luxemburg, March 27–April 1, 2021*, volume 12652 of *Lecture Notes in Computer Science*, pages 326–345. Springer, 2021. doi:10.1007/978-3-030-72013-1_17.
3. Pranav Ashok, Jan Kretínský, Kim Guldstrand Larsen, Adrien Le Coënt, Jakob Haahr Taankvist, and Maximilian Weininger. SOS: safe, optimal and small strategies for hybrid markov decision processes. In David Parker and Verena Wolf, editors, *Proceedings of the 16th International Conference on Quantitative Evaluation of Systems, QEST 2019, Glasgow, UK, September 10–12, 2019*, volume 11785 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2019. doi:10.1007/978-3-030-30281-8_9.
4. 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. doi:10.1016/0020-0190(79)90002-4.

5 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

6 Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

7 Nathalie Bertrand, Blaise Genest, and Hugo Gimbert. Qualitative determinacy and decidability of stochastic games with signals. *Journal of the ACM*, 64(5):33:1–33:48, 2017. doi:10.1145/3107926.

8 Dimitri P. Bertsekas and John N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991. doi:10.1287/moor.16.3.580.

9 Frantisek Blahoudek, Tomás Brázdil, Petr Novotný, Melkior Ornik, Pranay Thangeda, and Ufuk Topcu. Qualitative controller synthesis for consumption Markov decision processes. In Shuvendu K. Lahiri and Chao Wang, editors, *Proceedings (Part II) of the 32nd International Conference on Computer Aided Verification, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020*, volume 12225 of *Lecture Notes in Computer Science*, pages 421–447. Springer, 2020. doi:10.1007/978-3-030-53291-8_22.

10 Patricia Bouyer, Nathanaël Fijalkow, Mickael Randour, and Pierre Vandenhove. How to play optimally for regular objectives? In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10–14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 118:1–118:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.118.

11 Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. *Logical Methods in Computer Science*, 18(1), 2022. doi:10.46298/lmcs-18(1:11)2022.

12 Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-independent finite-memory determinacy in stochastic games. In Serge Haddad and Daniele Varacca, editors, *Proceedings of the 32nd International Conference on Concurrency Theory, CONCUR 2021, Virtual Conference, August 24–27, 2021*, volume 203 of *LIPIcs*, pages 26:1–26:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CONCUR.2021.26.

13 Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. The true colors of memory: A tour of chromatic-memory strategies in zero-sum games on graphs (invited talk). In Anuj Dawar and Venkatesan Guruswami, editors, *Proceedings of the 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, IIT Madras, Chennai, India, December 18–20, 2022*, volume 250 of *LIPIcs*, pages 3:1–3:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.3.

14 Romain Brenguier. *Equilibres de Nash dans les jeux concurrents : application aux jeux temporisés. (Nash equilibria in concurrent games : application to timed games)*. PhD thesis, École normale supérieure de Cachan, France, 2012. URL: https://tel.archives-ouvertes.fr/tel-00827027.

15 Léonard Brice, Jean-François Raskin, and Marie van den Bogaard. On the complexity of SPEs in parity games. In Florin Manea and Alex Simpson, editors, *Proceedings of the 30th EACSL Annual Conference on Computer Science Logic, CSL 2022, Göttingen, Germany, February 14–19, 2022*, volume 216 of *LIPIcs*, pages 10:1–10:17. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.10.

16 Thomas Brihaye, Véronique Bruyère, Aline Goeminne, and Jean-François Raskin. Constrained existence problem for weak subgame perfect equilibria with ω-regular Boolean objectives. *Information and Computation*, 278:104594, 2021. doi:10.1016/j.ic.2020.104594.

17 Thomas Brihaye, Véronique Bruyère, Aline Goeminne, Jean-François Raskin, and Marie van den Bogaard. The complexity of subgame perfect equilibria in quantitative reachability games. *Logical Methods in Computer Science*, 16(4), 2020. URL: https://lmcs.episciences.org/6883.

18 Thomas Brihaye, Véronique Bruyère, Aline Goeminne, and Nathan Thomasset. On relevant equilibria in reachability games. *Journal of Computer and System Sciences*, 119:211–230, 2021. doi:10.1016/j.jcss.2021.02.009.

19 Thomas Brihaye, Julie De Pril, and Sven Schewe. Multiplayer cost games with simple Nash equilibria. In Sergei N. Artëmov and Anil Nerode, editors, *Proceedings of the International Symposium on Logical Foundations of Computer Science, LFCS 2013, San Diego, CA, USA, January 6–8, 2013*, volume 7734 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2013. doi:10.1007/978-3-642-35722-0_5.

20 Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. To reach or not to reach? Efficient algorithms for total-payoff games. In Luca Aceto and David de Frutos-Escrig, editors, *Proceedings of the 26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1–4, 2015*, volume 42 of *LIPIcs*, pages 297–310. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.297.

21 Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*, 54(1):85–125, 2017. doi:10.1007/s00236-016-0276-z.

22 Thomas Brihaye and Aline Goeminne. Multi-weighted reachability games. *CoRR*, abs/2308.09625, 2023. doi:10.48550/arXiv.2308.09625.

23 Véronique Bruyère, Quentin Hautem, and Jean-François Raskin. On the complexity of heterogeneous multidimensional games. In Josée Desharnais and Radha Jagadeesan, editors, *Proceedings of the 27th International Conference on Concurrency Theory, CONCUR 2016, Québec City, Canada, August 23–26, 2016*, volume 59 of *LIPIcs*, pages 11:1–11:15. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.11.

24 Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Trading memory for randomness. In *Proceedings of the 1st International Conference on Quantitative Evaluation of Systems, QEST 2004, Enschede, The Netherlands, 27–30 September 2004*, pages 206–217. IEEE Computer Society, 2004. doi:10.1109/QEST.2004.1348035.

25 Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3-4):129–163, 2014. doi:10.1007/s00236-013-0182-6.

26 Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, STOC 1971, Shaker Heights, Ohio, USA, May 3–5, 1971*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047.

27 Costas Courcoubetis and Mihalis Yannakakis. Markov decision processes and regular events. *IEEE Transactions on Automatic Control*, 43(10):1399–1418, 1998. doi:10.1109/9.720497.

28 Wojciech Czerwiński and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7–10, 2022*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.

29 Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217, 2007. doi:10.1016/j.tcs.2007.07.008.

30 Florent Delgrange, Joost-Pieter Katoen, Tim Quatmann, and Mickael Randour. Simple strategies in multi-objective MDPs. In Armin Biere and David Parker, editors, *Proceedings (Part I) of the 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2020, Held as Part of ETAPS 2020, Dublin, Ireland, April 25–30, 2020*, volume 12078 of *Lecture Notes in Computer Science*, pages 346–364. Springer, 2020. doi:10.1007/978-3-030-45190-5_19.

31 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.

32 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 1997, Warsaw, Poland, June 29 – July 2, 1997*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

33 Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008. doi:10.2168/LMCS-4(4:8)2008.

34 Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. Games on graphs. *CoRR*, abs/2305.10546, 2023. doi:10.48550/arXiv.2305.10546.

35 Nathanaël Fijalkow and Florian Horn. The surprizing complexity of reachability games. *CoRR*, abs/1010.2420, 2010. doi:10.48550/arXiv.1010.2420.

36 Nathanaël Fijalkow and Florian Horn. Les jeux d'accessibilité généralisée. *Technique et Science Informatiques*, 32(9-10):931–949, 2013. doi:10.3166/tsi.32.931-949.

37 Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962. doi:10.1145/367766.368168.

38 Lester R. Ford. Network flow theory. Paper P-923, The RAND Corporation, Santa Monica, California, August 14, 1956. URL: http://www.rand.org/pubs/papers/P923.html.

39 James Friedman. A non-cooperative equilibrium for supergames. *Review of Economic Studies*, 38(1):1–12, 1971. URL: https://EconPapers.repec.org/RePEc:oup:restud:v:38:y:1971:i:1:p:1-12.

40 Drew Fudenberg and David Levine. Subgame-perfect equilibria of finite-and infinite-horizon games. *Journal of Economic Theory*, 31(2):251–268, 1983.

41 Marcus Gelderie. *Strategy machines: representation and complexity of strategies in infinite games*. PhD thesis, RWTH Aachen University, 2014. URL: http://darwin.bth.rwth-aachen.de/opus3/volltexte/2014/5025.

42 Hugo Gimbert and Wiesław Zielonka. Games where you can play optimally without any memory. In Martín Abadi and Luca de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory, CONCUR 2005, San Francisco, CA, USA, August 23–26, 2005*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2005. doi:10.1007/11539452_33.

43 Hugo Gimbert and Wiesław Zielonka. Pure and Stationary Optimal Strategies in Perfect-Information Stochastic Games with Global Preferences. Unpublished, 2009. URL: https://hal.archives-ouvertes.fr/hal-00438359.

44 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. doi:10.1007/3-540-36387-4.

45 Christoph Haase and Stefan Kiefer. The odds of staying on budget. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Proceedings (Part II) of the 42nd International Colloquium on Automata, Languages, and Programming, ICALP 2015, Kyoto, Japan, July 6–10, 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2015. doi:10.1007/978-3-662-47666-6_19.

46 Florian Horn. Random fruits on the Zielonka tree. In Susanne Albers and Jean-Yves Marion, editors, *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, Freiburg, Germany, February 26–28, 2009*, volume 3 of *LIPIcs*, pages 541–552. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, Germany, 2009. doi:10.4230/LIPIcs.STACS.2009.1848.

47 Rasmus Ibsen-Jensen. Concurrent games. In Nathanaël Fijalkow, editor, *Games on Graphs*. arXiv, 2023.

48 Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981. doi:10.1016/0022-0000(81)90039-8.

49 Neil D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975. doi:10.1016/S0022-0000(75)80050-X.

50 Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, Vladimir Gurvich, Gábor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008. doi:10.1007/s00224-007-9025-6.

51 Stephen Cole Kleene. Representation of events in nerve nets and finite automata. *C. E. Shannon and J. McCarthy (ed.). Automata Studies. Princeton University Press*, pages 3–42, 1956.

52 François Laroussinie, Nicolas Markey, and Ghassan Oreiby. Model-checking timed ATL for durational concurrent game structures. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2006, Paris, France, September 25–27, 2006*, volume 4202 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2006. doi:10.1007/11867340_18.

53 Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In Sumit Ganguly and Paritosh K. Pandya, editors, *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, Ahmedabad, India, December 11–13, 2018*, volume 122 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.38.

54 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7–10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

55 James C. A. Main. Arena-independent memory bounds for nash equilibria in reachability games. *CoRR*, abs/2310.02142, 2023. doi:10.48550/arXiv.2310.02142.

56 James C. A. Main and Mickael Randour. Different strokes in randomised strategies: Revisiting Kuhn's theorem under finite-memory assumptions. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *Proceedings of the 33rd International Conference on Concurrency Theory, CONCUR 2022, Warsaw, Poland, September 12–16, 2022*, volume 243 of *LIPIcs*, pages 22:1–22:18. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CONCUR.2022.22.

57 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., USA, 1967.

58 Benjamin Monmege, Julie Parreaux, and Pierre-Alain Reynier. Reaching your goal optimally by playing at random with no memory. In Igor Konnov and Laura Kovács, editors, *Proceedings of the 31st International Conference on Concurrency Theory, CONCUR 2020, Vienna, Austria, September 1–4, 2020*, volume 171 of *LIPIcs*, pages 26:1–26:21. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.26.

59 Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. The MIT Press, 1994.

60 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

61 Mickael Randour. Games with multiple objectives. In Nathanaël Fijalkow, editor, *Games on Graphs*. arXiv, 2023.

62 Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. *Formal methods in system design*, 50(2-3):207–248, 2017. doi:10.1007/s10703-016-0262-7.

63 Stéphane Le Roux and Arno Pauly. Extending finite-memory determinacy to multi-player games. *Information and Computation*, 261:676–694, 2018. doi:10.1016/j.ic.2018.02.024.

64 Bernard Roy. Transitivité et connexité. *Comptes Rendus de l'Académie des Sciences Paris*, 249:216–218, 1959.

65 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

66 Wolfgang Thomas. On the synthesis of strategies in infinite games. In Ernst W. Mayr and Claude Puech, editors, *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1995, Munich, Germany, March 2–4, 1995*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1995. doi:10.1007/3-540-59042-0_57.

67 Stephen D. Travers. The complexity of membership problems for circuits over sets of integers. *Theoretical Computer Science*, 369(1-3):211–229, 2006. doi:10.1016/j.tcs.2006.08.017.

68 Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. doi:10.1112/plms/s2-42.1.230.

69 Michael Ummels. Rational behaviour and strategy construction in infinite multiplayer games. In S. Arun-Kumar and Naveen Garg, editors, *Proceedings of the 26th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FST TCS 2006, Kolkata, India, December 13–15, 2006*, volume 4337 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2006. doi:10.1007/11944836_21.

70 Michael Ummels. The complexity of Nash equilibria in infinite multiplayer games. In Roberto M. Amadio, editor, *Proceedings of the 11th International Conference on Foundations of Software Science and Computational Structures, FoSSaCS 2008, Held as Part of ETAPS 2008, Budapest, Hungary, March 29 – April 6, 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 20–34. Springer, 2008. doi:10.1007/978-3-540-78499-9_3.

71 Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001. doi:10.1006/inco.2000.2894.

72 Stephen Warshall. A theorem on Boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962. doi:10.1145/321105.321107.

73 Avi Wigderson. The complexity of graph connectivity. In Ivan M. Havel and Václav Koubek, editors, *Proceedings of the 17th International Symposium on Mathematical Foundations of Computer Science, MFCS 1992, Prague, zechoslovakia, August 24–28, 1992*, volume 629 of *Lecture Notes in Computer Science*, pages 112–132. Springer, 1992. doi:10.1007/3-540-55808-X_10.

74 James Worrell. The Skolem landscape (invited talk). In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10–14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 5:1–5:2. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.5.

On Measuring Average Case Complexity via Sum-Of-Squares Degree

Prasad Raghavendra
University of California, Berkeley, CA, USA

Abstract

Sum-of-squares semidefinite programming hierarchy is a sequence of increasingly complex semidefinite programs to reason about systems of polynomial inequalities. The kth-level of the sum-of-squares SDP hierarchy is a semidefinite program that can be solved in time $n^{O(k)}$.

Sum-of-squares SDP hierarchies subsume fundamental algorithmic techniques such as linear programming and spectral methods. Many state-of-the-art algorithms for approximating NP-hard optimization problems are captured in the first few levels of the hierarchy. More recently, sum-of-squares SDPs have been applied extensively towards designing algorithms for average case problems. These include planted problems, random constraint satisfaction problems, and computational problems arising in statistics.

From the standpoint of complexity theory, sum-of-squares SDPs can be applied towards measuring the average-case hardness of a problem. Most natural optimization problems can often be shown to be solvable by degree n sum-of-squares SDP, which corresponds to an exponential time algorithm. The smallest degree of the sum-of-squares relaxation needed to solve a problem can be used as a measure of the computational complexity of the problem. This approach seems especially useful for understanding average-case complexity under natural distributions. For example, the sum-of-squares degree has been used to nearly characterize the computational complexity of refuting random CSPs as a function of the number of constraints.

Using the sum-of-squares degree as a proxy measure for average case complexity opens the door to formalizing certain computational phase transitions that have been conjectured for average case problems such as recovery in stochastic block models.

In this talk, we discuss applications of this approach to average-case complexity and present some open problems.

2012 ACM Subject Classification Theory of computation → Semidefinite programming; Theory of computation → Proof complexity; Theory of computation → Problems, reductions and completeness

Keywords and phrases semidefinite programming, sum-of-squares SDP, average case complexity, random SAT, stochastic block models

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.2

Category Invited Talk

Funding *Prasad Raghavendra*: Supported by NSF AF:2007676.

© Prasad Raghavendra;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 2; pp. 2:1–2:1
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Computational and Information-Theoretic Questions from Causal Inference

Leonard J. Schulman
California Institute of Technology, Pasadena, CA, USA

Abstract

Data, for the most part, is used in order to inform potential interventions: whether by individuals (decisions about education or employment), government (public health, environmental regulation, infrastructure investment) or business. The most common data analysis tools are those which identify correlations among variables – think of regression or of clustering. However, some famous paradoxes illustrate the futility of relying on correlations alone without a model for the causal relationships between variables.

Historically, causality has been teased apart from correlation through controlled experiments. But for a variety of reasons – cost, ethical constraints, or uniqueness of the system – we must often make do with passive observation alone. A theory based upon directed graphical models has been developed over the past three decades, which in some situations, enables statistically defensible causal inference even in the absence of controlled experiments.

Yet "some situations" is rather fewer than one would like. This limitation spurs a range of research questions. In this talk I will describe a couple of causality paradoxes along with how they are captured within the graphical model framework; this will lead naturally toward some of the computational and information-theoretic questions which arise in the theory.

2012 ACM Subject Classification Computing methodologies → Causal reasoning and diagnostics; Computing methodologies → Bayesian network models

Keywords and phrases Causal Inference, Bayesian Networks

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.3

Category Invited Talk

Funding *Leonard J. Schulman*: Research supported by NSF CCF-2321079.

© Leonard J. Schulman;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 3; pp. 3:1–3:1
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

From Concept Learning to SAT-Based Invariant Inference

Sharon Shoham
Tel Aviv University, Israel

Abstract

In recent years SAT-based invariant inference algorithms such as interpolation-based model checking and PDR/IC3 have proven to be extremely successful in practice. However, the essence of their practical success and their performance guarantees are far less understood. This talk surveys results that establish formal connections and distinctions between SAT-based invariant inference and exact concept learning with queries, showing that learning techniques and algorithms can clarify foundational questions, illuminate existing algorithms, and suggest new directions for efficient invariant inference.

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains; Theory of computation → Program verification; Software and its engineering → Formal methods

Keywords and phrases invariant inference, complexity, exact learning, interpolation, IC3

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.4

Category Invited Talk

Funding The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement No [759102-SVIS]).

© Sharon Shoham;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 4; pp. 4:1–4:1
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Algorithms in the Presence of Biased Inputs

Nisheeth K. Vishnoi
Yale University, New Haven, CT, USA

Abstract

Algorithms for optimization problems such as selection, ranking, and classification typically assume that the inputs are what they are promised to be. However, in several real-world applications of these problems, the input may contain systematic biases along socially salient attributes associated with inputs such as race, gender, or political opinion. Such biases can not only lead the outputs of the current algorithms to output sub-optimal solutions with respect to true inputs but may also adversely affect opportunities for individuals in disadvantaged socially salient groups. This talk will consider the question of using optimization to solve the aforementioned problems in the presence of biased inputs. It will start with models of biases in inputs and discuss alternate ways to design algorithms for the underlying problem that can mitigate the effects of biases by taking into account knowledge about biases. This talk is based on several joint works with a number of co-authors [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains; Theory of computation → Design and analysis of algorithms; Human-centered computing

Keywords and phrases Algorithmic Bias

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.5

Category Invited Talk

References

1. Niclas Boehmer, L. Elisa Celis, Lingxiao Huang, Anay Mehrotra, and Nisheeth K. Vishnoi. Subset selection based on multiple rankings in the presence of bias: Effectiveness of fairness constraints for multiwinner voting score functions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 2641–2688. PMLR, 2023. URL: `https://proceedings.mlr.press/v202/boehmer23a.html`.
2. L. Elisa Celis, Chris Hays, Anay Mehrotra, and Nisheeth K. Vishnoi. The effect of the rooney rule on implicit bias in the long term. In Madeleine Clare Elish, William Isaac, and Richard S. Zemel, editors, *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pages 678–689. ACM, 2021. `doi:10.1145/3442188.3445930`.
3. L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. Fair classification with noisy protected attributes: A framework with provable guarantees. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1349–1361. PMLR, 2021. URL: `http://proceedings.mlr.press/v139/celis21a.html`.
4. L. Elisa Celis, Amit Kumar, Anay Mehrotra, and Nisheeth K. Vishnoi. Bias in evaluation processes: An optimization-based approach. In *NeurIPS*, 2023.
5. L. Elisa Celis, Anay Mehrotra, and Nisheeth K. Vishnoi. Interventions for ranking in the presence of implicit bias. In Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna, editors, *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 369–380. ACM, 2020. `doi:10.1145/3351095.3372858`.

© Nisheeth K. Vishnoi;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 5; pp. 5:1–5:2
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

6 L. Elisa Celis, Anay Mehrotra, and Nisheeth K. Vishnoi. Fair classification with adversarial perturbations. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8158–8171, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/44e207aecc63505eb828d442de03f2e9-Abstract.html.

7 Lingxiao Huang and Nisheeth K. Vishnoi. Stable and fair classification. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2879–2890. PMLR, 2019. URL: http://proceedings.mlr.press/v97/huang19e.html.

8 Anay Mehrotra, Bary S. R. Pradelski, and Nisheeth K. Vishnoi. Selection in the presence of implicit bias: The advantage of intersectional constraints. In *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21–24, 2022*, pages 599–609. ACM, 2022. doi:10.1145/3531146.3533124.

9 Anay Mehrotra and Nisheeth K. Vishnoi. Fair ranking with noisy protected attributes. In *NeurIPS*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/cdd0640218a27e9e2c0e52e324e25db0-Abstract-Conference.html.

10 Anay Mehrotra and Nisheeth K. Vishnoi. Maximizing submodular functions for recommendation in the presence of biases. In Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben, editors, *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 – 4 May 2023*, pages 3625–3636. ACM, 2023. doi:10.1145/3543507.3583195.

Online Facility Location with Weights and Congestion

Arghya Chakraborty
Tata Institute of Fundamental Research, Mumbai, India

Rahul Vaze
Tata Institute of Fundamental Research, Mumbai, India

Abstract

The classic online facility location problem deals with finding the optimal set of facilities in an online fashion when demand requests arrive one at a time and facilities need to be opened to service these requests. In this work, we study two variants of the online facility location problem; (1) weighted requests and (2) congestion. Both of these variants are motivated by their applications to real life scenarios and the previously known results on online facility location cannot be directly adapted to analyse them.

Weighted requests: In this variant, each demand request is a pair (x, w) where x is the standard location of the demand while w is the corresponding weight of the request. The cost of servicing request (x, w) at facility F is $w \cdot d(x, F)$. For this variant, given n requests, we present an online algorithm attaining a competitive ratio of $\mathcal{O}(\log n)$ in the secretarial model for the weighted requests and show that it is optimal.

Congestion: The congestion variant considers the case when there is a congestion cost that grows with the number of requests served by each facility. For this variant, when the congestion cost is a monomial, we show that there exists an algorithm attaining a constant competitive ratio. This constant is a function of the exponent of the monomial and the facility opening cost but independent of the number of requests.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, online facility location, probabilistic method, weighted-requests, congestion

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.6

Related Version *arXiv Version*: `https://arxiv.org/abs/2211.11961`

Funding Research supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500.
Arghya Chakraborty: Research supported in part by the Google India Research Award (PI: Prahladh Harsha).

Acknowledgements The first author wishes to extend his deepest thanks to Prof. Umang Bhaskar, Prof. Prahladh Harsha, and Prof. Jaikumar Radhakrishnan for the thought-provoking discussions and the valuable advice they generously shared with him on multiple occasions.

1 Introduction

The facility location problem is one of the most well-studied problems in the field of algorithms [14, 3, 17], where Meyerson [18] studied the online version of this problem where requests arrive in a sequence and on arrival, the request must be allocated to a facility irrevocably and a cost is incurred depending on the distance of the request from the allocated facility. New facilities can also be opened on the arrival of a new request so that the current request and future requests have a nearby facility to be allocated to. However opening facilities incur a cost too and the goal is to minimize the total cost incurred.

© Arghya Chakraborty and Rahul Vaze;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 6; pp. 6:1–6:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To motivate the variants of online facility location we study, let us consider a plausible real-life scenario where such an online facility location algorithm may be deployed. Suppose the government is trying to set up vaccination centres in order to distribute vaccines depending on incoming requests. While online facility location can model the problem, a few other issues arise. For instance, one facility might not be able to satisfy a large number of requests and depending on the number of requests, there may be congestion. Also, there may be multiple vaccine doses resulting in weighted requests where each request needs to be satisfied a certain number of times (for instance, some people might require 2 doses while others require more[1]). There are numerous other real life applications of the facility location problem. These additional variants involving congestion and weighted-requests have their own costs which need to be taken into account when designing a solution and the classical facility location problem may not be directly applicable in these cases. In this work, we generalize the classical facility location problem to incorporate these extra variants and prove matching upper and lower bounds. We study two variants of the classical facility location, each incorporating one of these variants. We now define the classical online facility location problem and these two variants formally.

1.1 Online Facility Location Problem

The facility location problem comprises of a set of requests $\{x_1, x_2, ..., x_n\}$ on a metric space $(\mathcal{M}, d(\cdot\ ,\ \cdot))$. The objective is to open a set of facilities F on this metric space. There is a facility opening cost of f incurred for opening each such facility and a distance cost for each request which is proportional to the distance of the request to its nearest facility. Naturally, the objective is to open facilities such that the total cost incurred is minimum, where the total cost incurred is given by

$$f \cdot |F| + \sum_{i=1}^{n} d(x_i, F)$$

where $d(x_i, F)$ denotes the distance of x_i to its nearest facility in F. This problem is known to be NP-Hard [12].

The online variant of the facility location problem was first studied in detail by Meyerson [18]. In this setting, a request, x_i, needs to be allotted *on its arrival* to a facility, F, either an existing facility or a newly-opened facility. The cost incurred on allotting the request is the distance cost $d(x_i, F^{(i)})$ where $F^{(i)}$ refers to the set of facilities available at the time of servicing request x_i (including possibly the newly-opened facility). As in the offline setting, opening new facilities costs f for every facility opened. Meyerson [18] constructed a randomized algorithm, RFL (Randomized Facility Location), which attains a competitive ratio of $\mathcal{O}(\log n)$ in expectation. Later, Fotakis [9] proved that no online algorithm can attain a competitive ratio better than $\Omega\left(\frac{\log n}{\log\log n}\right)$ and also did an improved analysis proving that RFL attains a competitive ratio of $\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ in expectation.

Meyerson also showed that in the secretarial model (where the requests are adversarial but the arrival order is uniformly at random), his algorithm attains a competitive ratio of 8.

[1] We assume that these multiple requests must be served by the same facility else these multiple requests may be treated as independent requests.

1.2 The two variants and our results

Online facility location problem has many applications in real world. However, as observed by the vaccination example, some of these applications may have additional constraints on the classical online facility problem. In this paper, we study two variants incorporating these additional constraints.

1.2.1 Variant I: Weighted Requests

In our first variant, we consider a setting of weighted requests, where each request x_i arrives with an additional weight w_i. One may view this weight w_i as the number of times the request x_i needs to be served or simply as a weight corresponding to a premium client.

Formally, a sequence of requests (x_i, w_i) arrive in an online fashion where x_i is a point in the metric space $(\mathcal{M}, d(\cdot\ ,\ \cdot))$ while w_i is a positive real number. On arrival, the request (x_i, w_i) must be allocated to a facility, which could be one of the existing facilities or a newly-opened facility. As before, the facility opening cost is f but the cost incurred on allocating request (x_i, w_i) is $d(x_i, F^{(i)}) \cdot w_i$, instead of the usual $d(x_i, F^{(i)})$. If $w_i \cdot d(x_i, \cdot)$ is also a metric, this can be handled by the classical case, but this may not always be the case. As usual, the goal is to minimize the total cost incurred which is given by the following expression:

$$f.|F| + \sum_{i=1}^{n} d(x_i, F) \cdot w_i.$$

Note, that while our motivating example suggested that the weights w_i's are positive integers (corresponding to the number of times a request needs to be served), the above formulation is more general and allows for any positive real number.

We first modify Fotakis' lower bound to show that in the worst-case setting, no online algorithm can attain a competitive ratio better than $\Omega(n)$ matching the naïve algorithm that opens a new facility at every request. We then show that in the secretarial setting (when the requests are adversarial, but the ordering is random), there is an online algorithm that achieves a competitive ratio of $\mathcal{O}(\log n)$, which we show is tight once again by adapting Fotakis' lower bound. Observe that these results prove that this variant is provably different from the classical setting.

1.2.2 Variant II: Congestion

Our next variant incorporates the notion of congestion: if a facility is attending to multiple requests, there is an additional cost that it has to pay, depending on the number of requests it is attending to.

Formally, a sequence of requests x_i arrives and on arrival, the request must be allocated to a facility. Just as before, a facility may be opened in order to allocate x_i. A distance cost of $d(x_i, F^{(i)})$ is incurred and a facility opening cost of f is incurred every time a facility is opened. As of yet, this is exactly same as the online facility location problem. However we incorporate an additional congestion cost. If the facility to which x_i is allocated to has k requests allotted to it after x_i is allotted, then a congestion cost of $g(k) - g(k-1)$ is incurred in addition, where g is a convex non-decreasing function, satisfying $g(0) = 0$. We shall call g as the congestion function. In this model, if a total of ℓ requests are allocated to a facility, a total congestion cost of $g(\ell)$ is incurred and the goal is to minimize the total cost incurred. Note that the total cost includes three components: facility-opening costs, allocation costs and congestion costs.

In this work, we prove results for the special case when g is a monomial. We show that Meyerson's algorihtm, RFL (Randomized Facility Location), can be modified to obtain a $\mathcal{O}(\log k^* / \log\log k^*)$ competitive ratio, where $k^* = 2 \cdot g^{-1}\left(\frac{f}{g(2)-2}\right)$. Note that this is independent of n, the number of requests. We also show that this is tight up to a constant by providing a matching lower bound.

A few words on the restriction of congestion function to monomials. Our results work for any convex non-decreasing congestion function g that satisfies $g(0) = 0$ and $g(a \cdot b) = g(a) \cdot g(b)$ for all a, b. Hence, the assumption that g is a monomial. While the assumption that the congestion function g is a monomial is restrictive, it is interesting that in this setting, we get tight matching upper and lower bounds. It would be interesting to extend this proof to more general convex functions.

1.2.3 Proof Techniques

While all the algorithms that we analyse are modifications of Meyerson's algorithm RFL and the lower bound analyses are inspired by Fotakis' analysis, significant modifications are needed to handle each variant.

For the weighted requests, as indicated earlier, we show that in the adversarial model, no algorithm can attain a competitive ratio better than $\Omega(n)$, which is trivially achievable by opening a facility for all the requests. Then we analyse this in the secretarial model, where we are able to connect this problem to a problem on increasing sequences in a uniformly random permutation. We obtain both the upper and lower bound of $\Theta(\log n)$ by studying properties of increasing sequences in random permutations. In particular, we reduce the analysis of the algorithm to the analysis of a process related to permutations, which we refer to as the *Selection Process*, which might be of independent interest. The Selection Process has the property that if the input permutation is chosen adversarially, its cost can be as high as $\Omega(n)$ while on a random permutation, the expected cost drops to $\Theta(\log n)$.

For the model with congestion, we first show that the optimal offline algorithm (OPT) will not allocate more than k^* requests to a facility. Using this fact, we analyze any arbitrary facility c^* opened by OPT. We analyze the cost incurred by our online algorithm over the requests allocated to c^*. We are able to split up this cost carefully into two components – one component is simply the congestion cost while the other component has no congestion cost involved. Using the fact that the facility c^* has at most k^* requests allotted to it, we are able to attain a competitive ratio of $\mathcal{O}\left(\frac{\log k^*}{\log\log k^*}\right)$.

1.3 Related Work

Since many real-life problems are similar to facility location problems, it has received a lot of attention. It has even been used as a tool to solve other online problems (like [13, 6]) . The survey by Fotakis[10] gives a nice depiction of the problem and the general techniques that one needs to be familiar with. Other modifications that have been studied, include a setting where the facilities can themselves be moved with some cost [8], or the requests may be deleted [7]. The latter paper also studies capacitated facility location, which assumes that each facility can attend to at most a fixed number of requests. This is very similar to our congestion model, but as the bounds suggest, these two problems seem to be different.

Similarly, there has been interest on making these algorithms deterministic. Anagnostopoulos, Bent, Upfal, and Hentenryck in their paper [2] show that there is an $\mathcal{O}(\log n)$-competitive deterministic algorithm for online facility location. Later Fotakis in [10] showed

a deterministic primal dual algorithm attaining $\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ competitive ratio. More recently Kaplan, Naori, and Raz in [16] show that when the requests arrive uniformly at random, no online algorithm can achieve a competitive ratio better than 2 and a modification of Meyerson's algorithm achieves a competitive ratio of 3 in expectation.

Another interesting topic in online algorithms is to improve the performance based on advice. This has been studied widely in [1, 4, 11, 15] and indeed one can have algorithms with better competitive ratio if the advice is good.

One thing to note is that in all of these papers, the algorithms in most cases are modifications of the original algorithm by Meyerson. The analysis is what makes these problems interesting. This is true for our results as well where the algorithms are adaptations of Meyerson's algorithm however the analysis needs some novel ideas.

2 Facility Location with Weighted Requests

In this model, we assume that input requests are ordered pairs of the form (x_i, w_i) where x_i describes the position of the request while w_i is the weight of the request. This could be the model for how long the request stays in the system or the number of times the request has to be serviced. The corresponding cost incurred is $w_i \cdot d(x_i, F)$ (as opposed to just $d(x_i, F)$). Our results hold for arbitrary positive weight units, which need not be integers.

▶ **Definition 2.1** (Weighted Online Facility Location). *A sequence of n ordered pairs (x_i, w_i) are given as input, where $x_i \in \mathcal{M}$ for some metric space $\mathcal{M}$. Each such request needs to be allocated to a facility in an online fashion. Opening a facility incurs a cost of f and allocating a request (x_i, w_i) to a facility F incurs a cost of $d(x_i, F) \cdot w_i$. The goal is to open facilities and allocate the requests in an online fashion such that the total cost incurred is minimized.*

Here, the algorithm knows the metric space $\mathcal{M}$ but does not know n beforehand.

It is not hard to show that in the worst case setting, one can not expect to achieve a competitive ratio better than $\Omega(n)$ (See Section 4.2 for more details), which is trivially achievable by opening a facility on every request.

In light of the above, we study this setting in the secretarial model. Informally speaking, the secretarial model is one where the requests may be adversarial but the order in which they appear is uniformly at random. This ensures that the arrival order of the input requests cannot be adversarial. In other words, while the n input requests may be arbitrary, the secretarial model assumes that at each step all the remaining requests are equally likely to arrive as the next request. For the secretarial model, we show the following matching upper and lower bounds :

▶ **Theorem 2.2.** *In the online facility location problem with weighted requests, no online algorithm can obtain a competitive ratio better than $\Omega(\log n)$ in the secretarial model.*

▶ **Theorem 2.3.** *For the online facility location problem with weighted requests, there exists an algorithm attaining a competitive ratio of $\mathcal{O}(\log n)$ in expectation under the secretarial model.*

The proof of lower bound is deferred to Section 4 and the upper bound is proved in Section 5.

3 Facility Location with Congestion

In this model, we consider a sequence of n requests $x_1, ..., x_n$ from a metric space $(\mathcal{M}, d(\cdot,\cdot))$ endowed with a distance metric d. The goal is to open facilities when needed and assign these incoming requests to the facilities. We assume we are given a congestion function g, which is a convex non-decreasing function such that $g(0) = 0$. Opening a facility incurs a cost of f, while assigning a request x_i to a facility F containing k requests before x_i was assigned incurs a cost of $d(x_i, F) + g(k+1) - g(k)$. The additional $g(k+1) - g(k)$ is the congestion cost at the facility due to the new request.

▶ **Definition 3.1** (Online Facility Location with Congestion). *A sequence of n requests, x_i, is given as input, where $x_i \in \mathcal{M}$ for some metric space $\mathcal{M}$. Each such request needs to be allocated to a facility in an online fashion. Opening a facility incurs a cost of f and allocating a request x_i to a facility F containing k requests before x_i was assigned incurs a cost of $d(x_i, F) + g(k+1) - g(k)$. The goal is to open facilities and allocate the requests in an online fashion such that the cost incurred is minimized.*

In this paper, we consider congestion costs which satisfy $g(a \cdot b) = g(a) \cdot g(b)$ (for example a monomial). For this particular model, we prove the following :

▶ **Theorem 3.2.** *In the online facility location problem with congestion,*

1. *There exists an online algorithm attaining a competitive ratio of $\mathcal{O}(\frac{\log k^*}{\log\log k^*})$ in expectation, where $k^* := 2 \cdot g^{-1}\left(\frac{f}{g(2)-2}\right)$, a constant independent of the number of requests.*
2. *Furthermore, no randomized online algorithm can achieve a competitive ratio better than $\frac{\log k^*}{\log\log k^*}$.*

For want of space, the proofs of these theorems is deferred to Appendix B.

4 Lower bounds for the weighted-request variants

In this section we shall first show that in the worst case setting, no online algorithm can attain a competitive ratio better than $\Omega(n)$, which will be the motivation for us to consider the secretarial setting where we shall then show that no online algorithm may have a competitive ratio better than $\Omega(\log n)$ in expectation. This will make our analysis of Theorem 2.2 tight.

For the worst case setting, we shall work with a particular sequence of input requests. Then for the secretarial case, we shall use the same set of input requests assuming that the order of arrival of these requests is uniformly at random. Hence, we will first describe this set of requests.

4.1 The input for the lower bounds

The input requests will be constructed using a binary tree on a star metric. This is similar to the lower bound presented by Fotakis [9].

There will be requests along the nodes of the tree[2] in Figure 1, forming a path, starting from the root node and all the way to a leaf node. In order to refer to this tree we shall use the term "level". The root node is at level 0, the children of the root node are level 1 and for any node, its level is 1 added to the level of its parent node, all the way to the leaves being level $n-1$ nodes.

[2] The metric used for this analysis is the shortest path metric on the binary tree. However this can be embedded on the Euclidean Metric over $\mathbb{R}^1$. This was shown in [9].

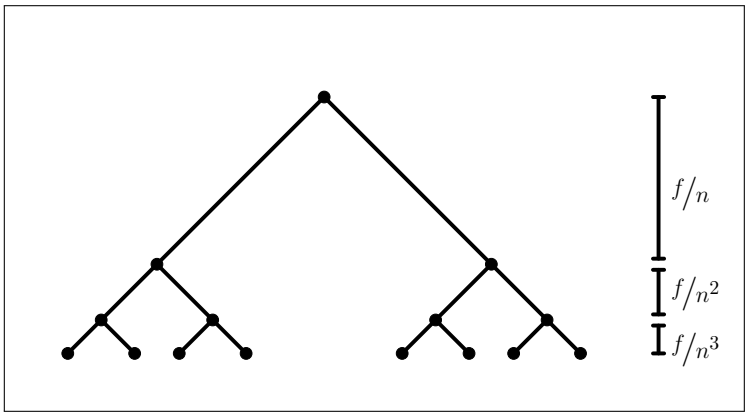


Figure 1 Tree of possible requests.

The tree will be such that the distance between consecutive nodes will keep decreasing exponentially: the distance between a node at level i and a node at level $i+1$ is $\frac{f}{n^{i+1}}$. We shall have 1 request at each node: the request at the i-th level node will have weight n^i.

For the worst case setting the requests will arrive in order of their levels - starting from the root node all the way to the leaf node. For the secretarial setting, the requests will be as mentioned but the arrival order will be uniformly at random.

4.2 Lower Bound in the Worst Case Setting

▶ **Proposition 4.1.** *No online algorithm can attain a competitive ratio better than* $\Omega(n)$.

Proof. For the mentioned input requests, an offline algorithm OFF may open 1 facility at the leaf node and in which case the distance of the root node to facility is at most $2\frac{f}{n}$ and for a node in the i-th level, its distance will be at most $2\frac{f}{n^{i+1}}$ from the leaf node. However taking into account the weight, the cost for the i-th level node is at most $2\frac{f}{n^{i+1}}.n^i = 2\frac{f}{n}$ Hence the total cost paid by OFF is less than $f + \sum_{i=0}^{n-1} 2\frac{f}{n} = f + 2f = 3f$.

Let us now consider the performance of an online algorithm on this input. At every level the algorithm must either

- Open a new facility
- Or pay a distance cost (taking into account the weight of the request) to its parent node at the very least

This is because, when opening a new facility, the online algorithm may either open a facility on the node of the current request (say v_i) or try to guess future nodes and open a facility somewhere in the subtree of the current level node request. Both of these will result in a new facility nonetheless. Now if the algorithm tries to guess and open a future node in the subtree (let us say left sub tree of v_i) instead of opening a facility at the current node, the adversary will select the next node for input request from the other subtree (right subtree of v_i) thus adversarially ensuring that guessing never gives the algorithm a correct future node.

This ensures that the algorithm does not have any available facility in the subtree of the current request. Hence, if the algorithm decides to pay a distance cost, on an i-th level request, instead of opening a facility anywhere, it must pay a distance cost at least to its parent, $\frac{f}{n^i}$ with weight n^i resulting in a cost of f. The other case consists of opening a facility and in that case the algorithm incurs a cost of f anyway. Therefore the algorithm incurs a cost of f for each level of the tree, resulting in a total cost of nf.

One needs to also consider the case where the algorithm opens multiple facilities on a requested node ensuring that both the subtrees have facilities in them. However in this case the algorithm has already paid the facility opening cost multiple times. If the algorithm opens k facilities on a requested i-th level node node (for $k \geq 2$), the algorithm cannot ensure that all the nodes in the $i + (k-1)$-th level below have facilities in their subtrees (since there are 2^{k-1} nodes at that level and $2^{k-1} \geq k$). This ensures that opening more than one facility on one node request is not beneficial for the algorithm.

Now let us compare the cost of the optimal offline algorithm, OPT, to the cost of an online algorithm. As we have already seen, the cost of any online algorithm is at least fn but the cost of OPT is at most $3f$ ensuring that $\frac{C_A}{C_{OPT}} \geq \frac{fn}{3f} = \frac{n}{3}$ for any online algorithm A.

Notice that this proof works against deterministic algorithms only. In order to prove this for randomized algorithms, we may use Yao's principle where as input we select one path among the 2^{n-1} paths uniformly at random. Alternatively, one can do this using the same technique (Lemma 4.5, in particular) used in Theorem 2.2, hence we have skipped the proof over here. ◀

A competitive ratio of n is trivially achievable by opening a facility on every request. However now, the naturally interesting question is whether an online algorithm can perform better if the input is secretarial.

4.3 Lower Bound in the Secretarial Setting

▶ **Definition 4.2.** *In the secretarial model, we assume that the adversary decides the input requests but not their arrival order. Instead after the adversary has decided the set of input requests, the arrival order of the requests is uniformly at random.*

Now, we shall give a proof for the lower bound in this setting.

Proof of Theorem 2.2. As mentioned earlier, we shall provide the same set of input requests but now the arrival order will be uniformly at random.

Notice that this makes the problem easier for an online algorithm in the sense that if an i-th level node is the first request to arrive, all the requests of level $0, 1, ..., i-1$ may be deduced by the online algorithm from the i-th level node and hence it will not have to open facilities for those requests when they arrive. Therefore we will not try to lower bound the cost of the algorithm on a request at level i if another request of higher level had already arrived earlier. However every time a node is requested such that no node below it has been requested yet, we shall ensure that the online algorithm has to pay some cost.

First we shall look at online algorithms, A, such that if it opens a facility on a request, it will open only at the location of the current request.

▷ Claim 4.3. Let A be an online algorithm such that on input requests, A would either open a facility at the request location or not open a facility at all. Then A cannot have a competitive ratio better than $\Omega(\log n)$.

Proof of Claim. As discussed earlier, the input will be composed in the following fashion :

- For an input of size n, we shall consider a binary tree on n levels : T_n.
- Then we shall choose a leaf node uniformly at random among the 2^{n-1} leaf nodes.
- There is a unique n length path connecting the root node to this leaf node.
- The input will comprise of requests on these nodes such that a request on the i-th level node will have weight of n^i.
- Also the distance between the $i-1$-th level node and the i-th level node will be $\frac{f}{n^i}$.

Notice that since we are in the secretarial model, these requested nodes (with corresponding weights) will arrive uniformly at random. Let us suppose that there are k requests $v_1, v_2, ..., v_k$ such that on arrival of the request v_i, it is the highest levelled node (or in other words, the lowest placed node in the tree structure) that has arrived till then.

Then for each of those requests, v_i, the online algorithm must either open a facility there or pay a distance cost with weight. Also, we have the guarantee that the nearest open facility available to the requested node is at least as far away as the distance to its parent node and hence when multiplied by the weight, the cost is exactly f, which is the same cost incurred for opening a facility. Therefore every time a request arrives such that no node of higher level has arrived yet, the online algorithm has to pay a cost of f. To finish off our proof we just need the following lemma:

▶ **Lemma 4.4.** *Let $S := [n]$ be the set containing the numbers 1 through n. Let π be a uniformly at random permutation of the set S. Let k_π be the number of times that the i-th element in the permutation π is the largest element observed till then. Then $\mathbb{E}_\pi[k_\pi] = \Theta(\log n)$.*

Proof. Let X_i be a random variable such that

$$X_i = \begin{cases} 1 & \text{if } \pi_i > \pi_j \text{ for all } j < i \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $\mathcal{P}[X_n = 1] = \frac{1}{n}$ since X_n will be 1 if and only if $\pi_n = n$ which has probability $\frac{1}{n}$. Also, conditioned on $\pi_{i+1}, \pi_{i+2}, ..., \pi_n$, the probability that $X_i = 1$ is $\frac{1}{i}$ since no matter what $\pi_{i+1}, ..., \pi_n$ are, π_i will be the largest among $\pi_1, \pi_2, ...\pi_i$ with probability $\frac{1}{i}$. Hence $\mathbb{E}[k_\pi] = \mathbb{E}[\sum X_i] = \sum \mathbb{E}[X_i] = \sum \frac{1}{i} = \Theta(\log n)$. ◀

This Lemma 4.4 gives us that in the secretarial input model, in expectation there will be $\Theta(\log n)$ requested nodes such that they were the highest levelled nodes on their arrival. Hence cost of online algorithm A would be $\Omega(f \cdot \log n)$ compared to the offline optimal algorithm whose cost is at most $3f$ as we had calculated earlier (Here we are using the fact that the cost of the offline algorithm is the same for the secretarial input model as was for the non-secretarial model). Therefore the competitive ratio of online algorithm A is $\Omega(\log n)$. ◁

Now we consider the case that an online algorithm may not only open facilities at requested nodes but it may also open a facility at a nearby location instead of opening the facility exactly on the request. This potentially allows the online algorithm to guess future nodes and maybe reduce the cost. We shall look at the same input that we worked with, in the Claim 4.3, and observe that such a guessing algorithm cannot do much better in expectation.

▶ **Lemma 4.5.** *Let T_n be a binary tree with a path chosen uniformly at random from the root node to a leaf node. If a node, v, of this path is revealed, and an algorithm A opens a facility in the subtree of v, the guess will match with the actual path on 2 nodes (apart from v) in expectation.*

Proof. Since the path was chosen uniformly at random, A can guess the next level node with probability 1/2. Also, 2 nodes can be guessed correctly by A with probability 1/4 and continuing in this fashion, the probability that A's guess matches with the path on i nodes is at most $\frac{1}{2^i}$.

Therefore the expected number of nodes that A can guess correctly by opening a facility is at most

$$S := \sum_{i=1}^{\infty} \frac{i}{2^i}.$$

Firstly, we can use Ratio Test to observe that this series is actually converging. Then we notice that

$$2S = \sum_{i=1}^{\infty} \frac{i}{2^{i-1}} = \sum_{i=0}^{\infty} \frac{i+1}{2^i} = 1 + \sum_{i=1}^{\infty} \frac{i+1}{2^i} = 1 + \sum_{i=1}^{\infty} \frac{i}{2^i} + \sum_{i=1}^{\infty} \frac{1}{2^i} = 1 + S + 1$$

$$\implies S = 2\,. \qquad \blacktriangleleft$$

This means that every time the algorithm A receives a node which is the highest level till then, it has the choice to pay a distance cost with weight, or open a facility at the request itself or opening a facility in the subtree of the node, guessing future nodes. We have already seen the competitive ratio for algorithms that do not guess future nodes. Let k be the number of nodes observed such that they were the highest levelled nodes seen on their arrival. Then we had seen that an algorithm that does not guess incurs a cost of f each time resulting in a cost of $k \cdot f$ and then we had observed that the expected value of k is of the order of $\log n$ to obtain the final competitive ratio.

Now however the algorithm may guess the future nodes and potentially incur a less cost. Let k be the number of nodes observed such that they were the highest levelled nodes seen on their arrival. Then we can show that A must incur a cost of $f \cdot \frac{k}{3}$ at least, in expectation. If A incurs a cost less than $f \cdot \frac{k}{3}$, then algorithm A has opened $< \frac{k}{3}$ facilities, and hence has $< \frac{k}{3}$ guesses. Therefore the total number of nodes that A has guessed correctly is $< 2 \cdot \frac{k}{3}$ in expectation (Using Lemma 4.5). Note that on the arrival of node, say v, even if the algorithm A might guess a node, say v_1, correctly (Where v_1 lies in the subtree of v), it may not reduce the cost incurred by A. This is because the next highest levelled node to arrive might be a child of v_1 in which case A needs to pay a cost of f anyway. In other words, correctly guessing a node might not reduce the number of highest levelled nodes observed, because the nodes guessed correctly might not be the highest levelled nodes to arrive. Therefore, taking into consideration the $< \frac{k}{3}$ nodes that A has paid for and the nodes that A has potentially guessed correctly, A has satisfied $< \frac{k}{3} + \frac{2k}{3} = k$ nodes in expectation (even after considering facilities opened, correctly guessed nodes and distance costs paid). This cannot be the case since A has to satisfy all the k nodes which were the highest levelled nodes on arrival.

Therefore A must incur a cost of $f \cdot \frac{k}{3}$ at least, in expectation. Since $k = \Theta(\log n)$ in expectation (Using Lemma 4.4), A must incur a cost of $\Omega(\log n)$ in expectation. $\blacktriangleleft$

5 Upper Bound for Weighted-Requests in the Secretarial Setting

Now we shall focus our attention on the setting with weighted requests and present the proof of Theorem 2.3.

In order to show this we will reduce the problem to a completely different problem, which we shall call *The Selection Process*. This is a novel idea and connecting the Weighted Facility Location Problem to the Selection Process is what allows us to get a tight upper bound of $\mathcal{O}(\log n)$.

▶ **Definition 5.1** (The Selection Process). *Let $S = [n]$. A sequence of (p_i, j_i) arrive in an online fashion where $p_i \in [0,1]^n$ is a vector of probabilities and $j_i \in S$ for $i \in [n]$. The first pair to arrive is (p_1, j_1) and we shall* ***select*** *the element j_1 with probability p_{1,j_1}.*

Similarly, on input (p_i, j_i), we shall select the element j_i with probability p_{i,j_i} but only if the element j_i is greater than the previously selected numbers.

Also, for all $1 \leq i \leq n-1$, and $j_i \in S$, we shall assume that $p_{i,j_i} \geq p_{i+1,j_i}$. The final quantity that we want to compute is the expected number of selected elements.

Let us first consider the case when the adversary selects both the p_i's and the j_i's. In this setting one can see that in the worst case, there can be as many as n elements selected. For this, consider the probability vectors p_i's to be all 1's throughout and the element arrival order be $j_1 = 1, j_2 = 2, ..., j_n = n$. In this case, all the elements will be selected.

Hence, let us consider the case when the adversary decides the probability vectors p_i's (satisfying the constraint of $p_{i,j_i} \geq p_{i+1,j_i}$) however the arrival order of the requests is uniformly at random. That is, we take a permutation, π of $[n]$ uniformly at random and at step i, the element appearing is π_i. In this secretarial setting, one can show that no matter what probabilities the adversary chooses, the number of selected elements is $\mathcal{O}(\log n)$ in expectation.

▷ Claim 5.2. The number of selected elements in The Selection Process is $\mathcal{O}(\log n)$ in expectation.

We shall prove Claim 5.2 later.

▶ Remark 5.3. Notice that the selection process does not necessarily choose the longest increasing subsequence since the longest increasing subsequence in a uniformly permuted array, is of the size of $\Theta(\sqrt{n})$. It is actually enough to observe that this is $\Omega(n)$, which follows from Erdős–Szekeres theorem [19].

On the other hand, Lemma 4.4 tells us that greedily selecting the largest element gives us an increasing subsequence of size $\Theta(\log n)$.

However the adversary may manipulate the probability vectors p_i's in a manner to make the largest increasing subsequences to be more likely to be picked. Indeed, in Remark 5.4, we show that if the probability vectors aren't enforced to be non-increasing, the adversary can make sure that there are $\Theta(\sqrt{n})$ selections in expectation. This is why the Claim 5.2 is of utmost importance.

▶ Remark 5.4. The fact that for all $j \in S$, and $1 \leq i \leq n-1$, we require $p_{i,j} \geq p_{i+1,j}$ is necessary. One may look at the example where the first $\sqrt{n}$ probability vectors are such that they have 1 in their first $\sqrt{n}$ terms and 0 everywhere else. The next $\sqrt{n}$ probability vectors have their first $\sqrt{n}$ terms 0, next $\sqrt{n}$ terms as 1 and all 0's after that. This continues for $\sqrt{n}$ many blocks of probability vectors, each with their corresponding block of 1's and remaining 0's. It is easy to see that given these probability vectors and a uniform arrival order of elements, one can expect $\Theta(\sqrt{n})$ elements to be selected in The Selection Process and $\mathcal{O}(\log n)$ would have been unachievable.

First, let us present our algorithm for facility location with weighted requests. Then we shall analyse the competitive ratio and see where The Selection Process comes in.

Algorithm 1 Weighted Randomized Facility Location (WRFL).

```
procedure WRFL(x_i, w_i)                    ▷ Input request is x_i with weight w_i
    F_i ← Nearest facility to x_i
    p_i ← min{1, d(x_i, F_i) × w_i / f}
    With probability p_i open a new facility at x_i and allocate x_i to it
    With probability 1 − p_i, assign x_i to F_i
end procedure
```

While algorithm $WRFL$ is a simple modification of Algorithm 2 (Meyerson's Algorithm RFL, which we describe later), the analysis is completely different. The analysis of RFL does not go through because of the weight component. This is also apparent from the fact that the competitive ratios are of different order : $\log n$ for $WRFL$ while $\frac{\log n}{\log \log n}$ for RFL. Hence the entire structure of the analysis is different.

Now, we shall prove Theorem 2.3 for the Algorithm 1 in particular.

Proof of Theorem 2.3. In order to analyse the algorithm, we shall divide the run of the algorithm into what we shall define as phases. However before that, we shall set up some notations and observe a few properties.

Let c^* be a facility opened by OPT and let S_{c^*} be the set of requests that OPT assigns to c^*. Also when we say that an algorithm incurs a cost of C over a subset of requests, say T, we will look at each request in the subset T and note how much the algorithm paid for that request when allocating it- be it distance cost or facility opening cost. For example, if the algorithm allocates the request to a facility which was already opened on a previous request, not from T, then we just take into account the distance cost without considering the facility opening cost for that facility.

We may focus our attention on one facility c^*, opened by OPT and look at the set of requests S_{c^*} that OPT assigns to c^*. It is easy to see that analysing one such cluster is sufficient for our analysis (an explanation is provided in the extended paper [5]). For the sake of convenience, we shall refer to this set of requests as S, whenever we mean S_{c^*}.

Now, in order to analyze this, we shall define the phases of the algorithm.

▶ **Definition 5.5.** *Corresponding to the facility c^*, opened by OPT, the run of the algorithm $WRFL$ can be divided into phases as follows :*

- *The algorithm is said to be in Phase 0 until $WRFL$ opens a facility, say F_1, on a request in S, at which point phase 1 starts.*
- *Phase 1 starts when $WRFL$ opens its first facility on a request of S (which we called F_1) and continues until $WRFL$ opens another facility, say F_2 on a request of S and this facility is closer to c^* than F_1.*

  ⋮

- *Phase i starts when $WRFL$ opens a facility F_i and continues until $WRFL$ opens another facility, say F_{i+1}, on a request of S and this facility, F_{i+1}, is closer to c^* than F_i.*

Observe that this definition of Phase is valid for the given cluster of requests S. With respect to another cluster, the phases would be different. Also, the algorithm does not know about the phases, it is only well defined during the analysis of the algorithm. Just to get a clear idea, we can note that the number of phases depends not only on which requests $WRFL$ opens a facility on, but it also depends on the arrival order of the requests.

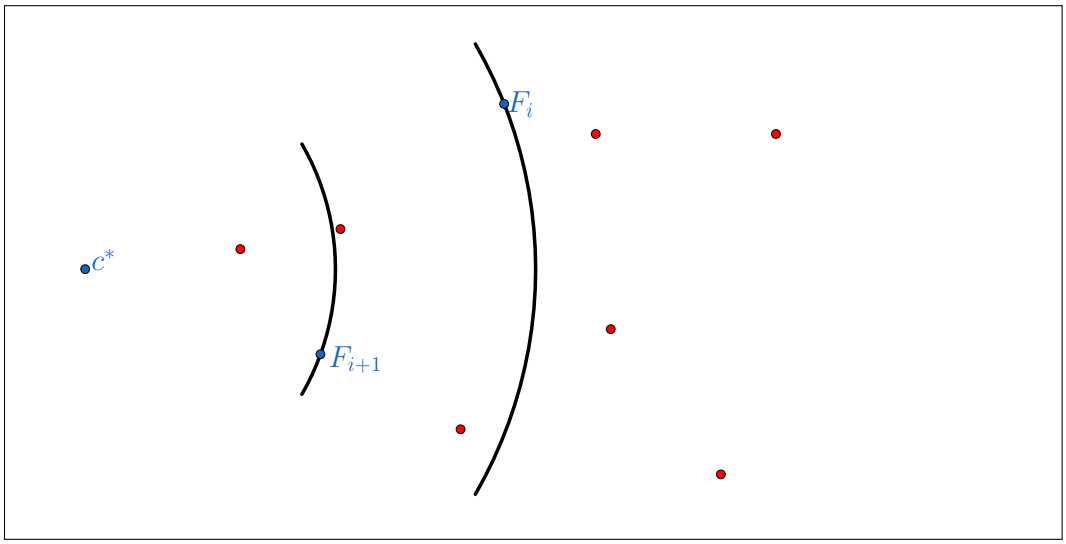


Figure 2 Phase i starts when Facility F_i is opened and ends when Facility F_{i+1} is opened where by definition F_{i+1} should be closer to c^* than F_i. All the requests in Phase i are shown in red dots. They may be closer or farther away from F_i with respect to c^*. The only guarantee that we have is that the first request closer than F_i that results in a facility opening would change the phase to $i+1$.

▶ **Lemma 5.6.** *Consider a set of requests, S, assigned to the same facility by OPT. Assume that the request arrivals and $WRFL$'s random choices are such that there are k phases. Then the cost ratio, for the requests in S, of $WRFL$ to OPT is $\mathcal{O}(k)$ in expectation.*

Proof. Let us first compute the cost of OPT over the requests in S. We note first that OPT has only opened one facility for these requests, at c^*. This constitutes a cost of f. Other than this, OPT incurs a cost of $C := \sum_{x_i \in S} d(x_i, c^*) \cdot w_i$.

To complete the proof, we shall use two properties in order to analyse the cost of $WRFL$.

▶ **Property 5.7.** *Let $X = \{(x_1, w_1), (x_2, w_2), \ldots, (x_m, w_m)\}$ be a subsequence of m requests. Then $WRFL$, in expectation, pays a cost $\leq f$ before it opens the first new facility in X.*

(This holds even if there are several open facilities at the beginning and even if new facilities are opened between two requests in the subsequence)

▶ **Property 5.8.** *On input request (x_i, w_i), let F be the nearest facility to x_i. Then the cost that $WRFL$ pays in expectation for the request (x_i, w_i) is at most $2d(x_i, F) \cdot w_i$.*

The proofs of these two Properties have been studied in earlier works (e.g. [9, 10]), but for the sake of completion we have added them in Appendix C.

Using Property 5.7, algorithm $WRFL$ would incur a cost of f in expectation before it opens a facility on a request of S. This would result in phase changing from Phase 0 to Phase 1. Now let us say that we are at Phase i and we want to estimate $WRFL$'s cost over the requests in Phase i, where Phase i started when $WRFL$ opened a facility F_i. Here F_i is closer than F_{i-1} to c^*, by construction.

Type 1 requests. Let S_1 be the set of requests such that $d(c^*, x) < d(c^*, F_i)$ for all $x \in S_1$. Then using Property 5.7, algorithm $WRFL$ pays a cost of f in expectation over requests in S_1 before it opens a facility F_{i+1} on a request with $d(c^*, F_{i+1}) < d(c^*, F_i)$. This event of opening a facility ends up changing the phase from i to $i+1$. Hence for phase i, requests closer to c^* than F_i incur a cost of at most f in expectation.

Type 2 requests. Let S_2 be the set of requests such that $d(c^*, x) \geq d(c^*, F_i)$ for all $x \in S_2$. Now observe that OPT incurs a cost of $d(c^*, x) \cdot w_x$ for the request x (Where w_x denotes the weight of the request x). Let us now try to estimate the cost incurred by $WRFL$ for this request x. Here, $WRFL$ has an open facility F_i and $d(F_i, x) \leq d(x, c^*) + d(c^*, F_i)$ using triangle inequality. Also, $d(c^*, x) \geq d(c^*, F_i)$ by assumption. So $d(F_i, x) \leq d(x, c^*) + d(x, c^*) = 2d(x, c^*)$. Therefore the nearest facility to x for $WRFL$ is at most at a distance of $2d(x, c^*)$. Hence by Property 5.8, the cost incurred by $WRFL$ for the request x is at most $4d(x, c^*) \times w_x$ in expectation, which is 4 times the cost incurred by OPT itself.

Hence the total cost of Type 2 requests for $WRFL$ is at most $4C$ in expectation (Where OPT had incurred a cost of $f + C$ for the requests in S). On the other hand, $WRFL$ incurs a cost of f for Type 1 requests in expectation, for each phase. Also $WRFL$ incurs a facility opening cost of f for each phase. Therefore the total cost incurred by $WRFL$ is at most $2k \cdot f + 4C$ in expectation. This results in an expected competitive ratio of $2k + 4 = \mathcal{O}(k)$. ◀

▶ **Lemma 5.9.** *The expected number of phases is $\mathcal{O}(\log n)$.*

Proof. In order to view this we will reduce the problem to The Selection Process (Definition 5.1), which we had mentioned earlier.

We shall also assume the Claim 5.2 for now and finish the proof of Lemma 5.9, after which we shall give a formal proof of Claim 5.2. Let the requests in S be ordered in decreasing distance from c^* and labelled accordingly. Thus x_1 is the request farthest from c^*, x_2 is the second farthest request and so on till x_n which is the request nearest to c^*.

Just before the first request from S arrives online, there may be open facilities due to other requests. Given the configuration of the facilities (opened by $WRFL$) at this stage, each request $x_i \in S$ has a certain probability of opening a facility if it were the first request to arrive in S. This vector of probabilities will be our p_1 with $p_{1,i}$ being the probability for x_i opening a facility, if it was the first request to arrive. Now one of the requests x_{π_1}, arrives uniformly at random. The index of the request arriving corresponds to the element arriving in The Selection Process. Note that the vector of probabilities thus generated, depends on the previous requests and whether these requests have opened new facilities or not. However, this still adheres to the Selection Process, as long as the requests arrive uniformly at random.

At any stage when the i-th request x_{π_i} arrives, we can similarly compute the vector p_i and also this request would change the phase only if a facility is opened there and also if it is nearer to c^* than all the other opened facilities, which corresponds to the index of the request being larger than the indices of all the opened facilities. So, any time the phase changes, a facility is opened which is nearer to c^* than all previously opened facilities. This corresponds to the element being selected in the Selection Process.

Also while these probability vectors may be arbitrary, the probability of opening a facility on a particular request can only decrease over time. This is because if the request arrives later on, it can potentially have nearby facilities, reducing the probability of opening a facility but this probability of opening a facility cannot increase with time, implying $p_{i,j} \geq p_{i+1,j}$.

Therefore, if we can show that the number of selected elements in The Selection Process is $\mathcal{O}(\log n)$ in expectation, then we would have shown that the number of phases is $\mathcal{O}(\log n)$ in expectation, as was required. ◀

Using Lemmas 5.6 and 5.9, it follows that the competitive ratio of $WRFL$ is $\mathcal{O}(\log n)$. ◀

Proof of Claim 5.2. In order to prove this, we shall reduce the problem one more time.

Notice that the i-th element to arrive is π_i and is selected with probability p_{i,π_i}, if no larger element has yet been selected. We can view the selection probability of π_i as

$$p_{i,\pi_i} = p_{1,\pi_i} \times \frac{p_{2,\pi_i}}{p_{1,\pi_i}} \times \frac{p_{3,\pi_i}}{p_{2,\pi_i}} \times ... \times \frac{p_{i,\pi_i}}{p_{i-1,\pi_i}} .$$

Since $p_{j,\ell} \geq p_{j+1,\ell}$ for all j (by definition), each of the terms $\frac{p_{j+1,\pi_i}}{p_{j,\pi_i}}$ is less than or equal to 1. Hence we can view the terms, $\frac{p_{j+1,\pi_i}}{p_{j,\pi_i}}$, as probabilities.

Now just before the first element π_1 arrives, we can toss random coins for each i such that the i-th coin is 1 with probability $p_{1,i}$. Given that π_1 is the first element to arrive, we can look at the realization of p_{1,π_1} and select it if it is 1.

Now just before the second element arrives, we can toss random coins for each i such that the i-th coin is 1 with probability $\frac{p_{2,i}}{p_{1,i}}$. Now, after observing π_2, the second element to arrive, we can look at the realizations of p_{1,π_2} and $\frac{p_{2,\pi_2}}{p_{1,\pi_2}}$ and select the element π_2 only if both the entries are 1, and additionally if $\pi_2 > \pi_1$. Notice that π_2 is selected with probability p_{2,π_2} in this manner, as needed.

Similarly this process continues for the i-th element arriving, π_i, for all $i \geq 3$. Here we toss random coins for each element i that are 1 with probability $\frac{p_{i,\pi_i}}{p_{i-1,\pi_i}}$ and we take into account the realizations of the previous coin tosses of probability $\frac{p_{i',\pi_i}}{p_{i'-1,\pi_i}}$ (for all $i' < i$). The element π_i is selected only if all of the realizations are 1 and π_i is larger than all selected elements. One can see that selecting the elements in this manner produces the same probability of an element being selected.

However now we can state that an element π_i that has not yet arrived will certainly not be selected on its arrival if even one of the realizations $\frac{p_{i',\pi_i}}{p_{i'-1,\pi_i}}$ or p_{1,π_i} happens to be 0 (where $i' < i$, since π_i has not yet arrived), also we can say that the element π_i will certainly not be selected on its arrival if an element larger than π_i has already been selected.

Let us take one such array $A[.][.]$ of realizations of the probabilities. Where $A[i][j]$ is 1 with probability $\frac{p_{i,j}}{p_{i-1,j}}$. On the arrival of an element, π_i, we look at the π_i-th column and select the element π_i only if all the first i entries of the column are 1. Hence once a 0 appears in a column, that element will certainly not be selected and instead of tossing further coins for the element, we shall set all the remaining entries in the column to be 0. This maneuver does not alter the probability of any arriving element being selected. Also if a larger element had already been selected, the corresponding element will certainly not be selected.

▷ Claim 5.10. Let i be the first index when selection happens and π_i be the first element to be selected. Then either $\mathbb{E}[\pi_i] \geq \frac{n}{3}$ or the i-th row of A has at least $\frac{n}{3}$ entries as 0.

Proof. Let the i-th row of A have at least $\frac{2n}{3}$ entries as 1. This means that the columns corresponding to those $\frac{2n}{3}$ entries have 1 throughout, for the first i entries. We shall then conclude that $\mathbb{E}[\pi_i] \geq \frac{n}{3}$ to complete this proof.

Let $c_1, c_2, ..., c_k$ be the columns of A with 1's throughout till the i-th row. By our assumption, $k \geq \frac{2n}{3}$. Under the condition that the first element to be selected is the i-th element, the expected value of the selected element equals

$$\mathbb{E}[\pi_i] \overset{(1)}{=} \frac{1}{k}\sum_{j=1}^{k} c_j \overset{(2)}{\geq} \frac{1}{k}\sum_{j=1}^{k} j = \frac{k+1}{2} \geq \frac{\frac{2n}{3}+1}{2} \geq \frac{n}{3}.$$

Outcome of coin toss of ith iteration $\longrightarrow$ \ Element $j \longrightarrow$								
p_1	1	1	1	1	0	1	1	1
$\frac{p_2}{p_1}$	0	1	1	1		1	0	1
$\frac{p_3}{p_2}$		1	0	1		0		1
$\frac{p_4}{p_3}$		1		0				1
$\frac{p_5}{p_4}$		1						0
		0		$\frac{p_{i,j}}{p_{i-1,j}}$				
$\frac{p_n}{p_{n-1}}$								

Figure 3 Array A of probabilities.

Equality (1) follows from the fact that because of the secretarial model, all the elements c_j are equally likely to be the i-th element. Since the i-th element, π_i, is the first selected element, π_i has to be one of the elements $\{c_1, c_2, ..., c_k\}$ only. Also none of the elements c_j have appeared earlier, else they would have been selected first. Hence all of those k elements are equally likely to be π_i.

Inequality (2) holds because all the k elements are distinct integers from $[n]$. Therefore the sum will be the smallest when the k elements are $1, 2, ..., k$.

Hence we have that if the i-th row of A has at least $\frac{2n}{3}$ entries as 1 then $\mathbb{E}[\pi_i] \geq \frac{n}{3}$, which proves our claim. $\triangleleft$

Applying Claim 5.10 recursively completes the proof. When the first element is selected, in expectation there will be $\frac{n}{3}$ elements that cannot be selected - either because they are smaller than the element picked or because they already have 0's in their columns in A. Therefore we can *discard* all those elements that can no longer be selected - meaning, if element j can no longer be selected we may remove the j-th row and the j-th column from the matrix A. Now we will have a smaller matrix, $A_1[.][.]$, leaving apart the rows and columns of the discarded elements. On this matrix, A_1, we shall again apply the Claim 5.10 to discard $\frac{1}{3}$ fraction of its elements in expectation, when the next element is selected. This process continues and Claim 5.10 states that each time an element is selected, $\frac{1}{3}$rd of the remaining elements are discarded in expectation. Hence this process ends in $\mathcal{O}(\log n)$ many steps in expectation. In other words, the number of elements selected is $\mathcal{O}(\log n)$. $\triangleleft$

References

1 Matteo Almanza, Flavio Chierichetti, Silvio Lattanzi, Alessandro Panconesi, and Giuseppe Re. Online facility location with multiple advice. In *Advances in Neural Information Processing Systems*, volume 34, pages 4661–4673. Curran Associates, Inc., 2021. URL: `https://proceedings.neurips.cc/paper/2021/file/250473494b245120a7eaf8b2e6b1f17c-Paper.pdf`.

2 Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Inf. Comput.*, 194(2):175–202, 2004. `doi:10.1016/j.ic.2004.06.002`.

3 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 106–113, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276718.

4 Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 35–66, 2022. doi:10.1137/1.9781611977073.3.

5 Arghya Chakraborty and Rahul Vaze. Online facility location with timed-requests and congestion, 2022. arXiv:2211.11961.

6 Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 30–39, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/780542.780548.

7 Marek Cygan, Artur Czumaj, Marcin Mucha, and Piotr Sankowski. Online Facility Location with Deletions. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2018.21.

8 Björn Feldkord and Friedhelm Meyer auf der Heide. Online facility location with mobile facilities. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 373–381. ACM, 2018. doi:10.1145/3210377.3210389.

9 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, January 2008. doi:10.1007/s00453-007-9049-y.

10 Dimitris Fotakis. Online and incremental algorithms for facility location. *SIGACT News*, 42(1):97–131, 2011. doi:10.1145/1959045.1959065.

11 Dimitris Fotakis, Evangelia Gergatsouli, Themis Gouleakis, and Nikolas Patris. Learning augmented online facility location. (manuscript), 2021. arXiv:2107.08277.

12 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.

13 Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528, March 2003. doi:10.1109/TKDE.2003.1198387.

14 Kamal Jain and Vijay Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of The ACM - JACM*, 48, March 2001. doi:10.1145/375827.375845.

15 Shaofeng H.-C. Jiang, Erzhi Liu, You Lyu, Zhihao Gavin Tang, and Yubo Zhang. Online facility location with predictions. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. arXiv:2110.08840.

16 Haim Kaplan, David Naori, and Danny Raz. Almost tight bounds for online facility location in the random-order model. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1523–1544. SIAM, 2023. doi:10.1137/1.9781611977554.ch56.

17 Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992. doi:10.1016/0020-0190(92)90208-D.

18 Adam Meyerson. Online facility location. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001. doi:10.1109/SFCS.2001.959917.

19 Andrew Suk and Ji Zeng. A Positive Fraction Erdős-Szekeres Theorem and Its Applications. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.62.

A Randomized Algorithm for Online Facility Location

Here, we mention the algorithm, RFL, as suggested by Meyerson[18], to illustrate the motivation behind our algorithms.

Algorithm 2 Randomized Facility Location (RFL).

```
procedure RFL(x_i)                                   ▷ Input request is x_i
    F_i ← Nearest facility to x_i
    p_i ← min{1, d(x_i, F_i)/f}
    With probability p_i open a new facility at x_i and allocate x_i to it
    With probability 1 − p_i, assign x_i to F_i
end procedure
```

B Facility Location with Congestion (with proofs)

In this section, we restate our results on congestion and provide the necessary proofs. First, let us recall Definition 3.1, where we defined Online Facility Location with Congestion. Also, note that in this paper, we consider congestion costs which satisfy $g(a \cdot b) = g(a) \cdot g(b)$ (for example a monomial). For this particular model, we prove the following :

▶ **Theorem B.1.** *In the online facility location problem with congestion,*

1. *There exists an online algorithm attaining a competitive ratio of* $\mathcal{O}(\frac{\log k^*}{\log \log k^*})$ *in expectation, where* $k^* := 2 \cdot g^{-1}\left(\frac{f}{g(2)-2}\right)$*, a constant independent of the number of requests.*
2. *Furthermore, no randomized online algorithm can achieve a competitive ratio better than* $\frac{\log k^*}{\log \log k^*}$.

In the rest of this section, we first prove the Lower Bound for the Worst Case input in Appendix B.1. Then we prove that the same bound is achieved, in expectation, by a randomized algorithm in Appendix B.2. However before we do these, we shall make a key observation that in the congestion model, an offline optimal algorithm will not allocate too many requests to a single facility.

Let k' be the smallest integer such that $g(k'+1) - g(k') \geq f$. If an online algorithm allocates $k'+1$ requests to one facility then the congestion cost for the $k'+1$-th request itself is $g(k'+1) - g(k') \geq f$. So no online algorithm needs to put more than k' requests in a facility as it might as well open a new facility and incur less cost.

The above reasoning holds for offline algorithms too and specifically for the offline optimal algorithm, OPT. However we can give an even better bound when considering OPT. For this we shall assume that the congestion function satisfies $g(a \cdot b) = g(a) \cdot g(b)$.

▷ Claim B.2. Any facility opened by OPT has at most $k^* := 2 \cdot g^{-1}\left(\frac{f}{g(2)-2}\right)$ requests allotted to it.

Proof. Let F be a facility opened by OPT with k requests in it. Since OPT did not split the k requests into two facilities but kept them in a single facility, it must be the case that splitting them into two facilities costs more. Notice that the cost paid if two facilities are opened is $\geq 2f + g(a_1) + g(a_2)$ in addition to some distance cost, where $a_1 + a_2 = k$. Since g is a convex non-decreasing function, we have $g(a_1) + g(a_2) \geq 2g\left(\frac{k}{2}\right)$ (Here we have assumed that k is even, else we can look at $k-1$ instead of k). Also opening two facilities means that the total distance cost is lesser compared to opening only one facility. Hence we can conclude $2f + 2g\left(\frac{k}{2}\right) \geq f + g(k)$. Since we are working under the assumption that $g(a \cdot b) = g(a) \cdot g(b)$, we can now write $f \geq (g(2) - 2)g\left(\frac{k}{2}\right) \implies \frac{k}{2} \leq g^{-1}\left(\frac{f}{g(2)-2}\right) \implies k \leq 2.g^{-1}\left(\frac{f}{g(2)-2}\right)$. ◁

B.1 Lower Bound

Here we shall present a proof of Theorem B.1-Item 2. The proof will be along the same lines as the previous lower bound proof.

Proof of Theorem B.1-Item 2. Consider a binary tree of depth h. The distance between root node and its children is $\frac{f}{h}$ but the distance keeps decreasing as we traverse down the tree. The distance between any level i node and its child will be $\frac{f}{h^{i+1}}$, where we will decide later what h to choose. The input will comprise of 1 request at root node, followed by h requests at a node at level 1, and so on, with h^i requests for some node at the ith level. The requested nodes will form a path from the root node to a leaf node, say ℓ. Also there will be a total of $n = 1 + h + h^2 + ... + h^{h-1} = \frac{h^h-1}{h-1}$ requests.

One may use this sequence of requests to obtain the required lower bound. Due to lack of space, the proof has been skipped here. The full version of the paper [5] contains the complete proof. ◀

B.2 Upper Bound

We have already seen a lower bound of the online facility location problem with congestion. Now we will present an algorithm to solve the online facility location problem with congestion and study the upper bound. We shall present an algorithm that asymptotically attains the same competitive ratio as suggested by the lower bound, making our analysis tight.

We shall present this algorithm as a modified version of RFL. Then we shall study the modified algorithm and try to compare the competitive ratio of the modified algorithm with congestion to the competitive ratio of the original algorithm without congestion.

However notice that the lower bounds for the competitive ratio is different for the case with congestion as compared to the original case without congestion. As we have seen earlier, the lower bound for the competitive ratio is $\frac{\log k^*}{\log\log k^*}$, when we are looking at online facility location with congestion. However in the original setting without congestion, the lower bound was $\frac{\log n}{\log\log n}$, which as we have seen is attained by RFL in expectation. Intuitively, the reason why we will be able to go from n to k^* is because in the congestion model we have the guarantee that OPT will not have more than k^* requests allocated to any facility so if we take one facility opened by OPT and consider the requests allotted to it, we can expect to somehow get a competitive ratio of $\frac{\log k^*}{\log\log k^*}$. After that we will just have to show that the competitive ratio holds true even when there are multiple facilities opened by OPT.

We shall use the algorithm RFL as a black box in order to obtain this modified algorithm, which we shall name $MRFL$.

Algorithm 3 Modified RFL (MRFL).

```
procedure MRFL(x_i)                                   ▷ Input request is x_i
    On input request x_i, run algorithm RFL on the current state for the input x_i and
    perform the same action.
    if Any facility, F, has k* requests allocated to it then
        "Close" facility F
        Open a new facility at the same location as F
    end if
end procedure
```

We shall now prove Theorem B.1-Item 1, in particular for the Algorithm $MRFL$.

Proof of Theorem B.1-Item 1. Firstly, by closing a facility, we mean that no more requests will be allocated to the facility anymore by the algorithm. No change in the model is assumed since the facility still exists: just that the algorithm will not allocate any more requests to it. Since another facility is opened right after a facility closes, at the same exact location, for future queries the algorithm will have this new facility available. Therefore the actions taken by RFL in the model without congestion (For e.g., allocating a request to a facility which has now become "closed") can always be performed by $MRFL$ in the model with congestion (Since the "closed" facility has another facility open at the exact same location). Also apart from the extra facilities opened by $MRFL$ in step 5, the actions taken by RFL and $MRFL$ are identical. This means that the distance cost paid by the algorithm RFL on a model without congestion is the same as the distance cost paid by the algorithm $MRFL$ on a model with congestion whenever the sequence of input requests is the same. We cannot say the same for the facility opening cost though, since the algorithm $MRFL$ opens excess facilities in step 5. However if we separate the facilities opened by $MRFL$ into two categories : facilities opened by $MRFL$ in step 2 and facilities opened by $MRFL$ in step 5, we will notice that the number of facilities opened by $MRFL$ in step 2 is equal to the number of facilities that RFL would have opened in the congestion free model for the same sequence of input requests. Visually, the total number of points in the metric space where a facility is opened is same for $MRFL$ and RFL, just that $MRFL$ might have multiple facilities opened at a few points in the metric space. Also the congestion cost is something that only $MRFL$ has to pay and RFL does not need to pay since RFL runs on a congestion free model.

Based on this observation, we shall now try to divide the total cost paid by $MRFL$ into two parts. Let C be the total cost paid by $MRFL$ over the entire sequence of input requests. Let C_1 be the total distance cost paid by $MRFL$ plus the total facility opening cost paid by $MRFL$ for facilities opened in step 2 of the algorithm. Let C_2 be the total congestion cost paid by $MRFL$ plus the total facility opening cost paid over all facilities opened in step 5 of the algorithm.

Clearly $C = C_1 + C_2$ and hence we can write the competitive ratio of algorithm $MRFL$ as

$$\mu = \frac{C}{C_{OPT}} = \frac{C_1 + C_2}{C_{OPT}} \ .$$

where C_{OPT} is the cost paid by an optimal offline algorithm on the same sequence of input requests.

▷ Claim B.3. If the total number of input requests is n then the cost C_2 incurred by algorithm $MRFL$ is at most $\frac{n}{k^*} f \left(1 + \frac{g(2)}{g(2)-2}\right)$.

Due to lack of space, the proof of this claim has been omitted. It can be found in the extended version of this paper [5].

At this point, using Claim B.2, we can say the OPT must open at least $\frac{n}{k^*}$ facilities and hence $C_{OPT} \geq \frac{n}{k^*} f$. Also using Claim B.3, we know that $C_2 \leq \frac{n}{k^*} f \left(1 + \frac{g(2)}{g(2)-2}\right)$. Therefore,

$$\frac{C_2}{C_{OPT}} \leq 1 + \frac{g(2)}{g(2) - 2} \ . \tag{1}$$

Hence we can focus our attention on $\frac{C1}{C_{OPT}}$, since $\frac{C_2}{C_{OPT}}$ is a constant. Let us first try to understand what these terms represent. C_1 represents the total distance cost paid by the algorithm $MRFL$ plus the facility opening cost for facilities opened in step 2 of the Algorithm 3. We notice that the distance cost paid by $MRFL$ over a sequence of input requests is equal to the distance cost paid by RFL over the same sequence of input requests

in the model without congestion. Also, the number of facilities opened by RFL is the same as the number of facilities opened by $MRFL$ in step 2 by construction. Hence C_1, in expectation, is the cost paid by RFL on the same sequence of input requests in the model without congestion.

However the denominator C_{OPT} is the cost paid by OPT for the same sequence of input requests in the model with congestion. One thing to note here is that if the denominator was the cost paid by OPT without congestion, we would have been stuck because then this ratio would have been $\mu(n) := \frac{\log n}{\log \log n}$, which is much larger than the competitive ratio of $\frac{\log k^*}{\log \log k^*}$ we are trying to achieve.

At this point we would have ideally liked to say that since OPT allocates at most k^* requests to any facility, let us take any particular facility c^* opened by OPT. If we looked only at the requests allotted to c^* by OPT, the cost paid by algorithm RFL is at most $\mu(k^*)$ times the cost paid by OPT had the input been only these requests allotted to c^* by OPT. However the input actually comprises of several such clusters, where each cluster is the set of requests that were allocated to the same facility by OPT. While it is true that the competitive ratio would have been $\mu(k^*)$ if the input was only one such cluster, it is not clear if that will be the case for the entire input sequence. This is because while OPT's cost over the entire input sequence is exactly the sum of costs of the clusters, RFL might end up paying more on the entire input when compared to RFL's costs on the clusters individually (had the input been just the cluster), summed up.

More formally, let us view the entire input sequence of length n as clusters of input sequences where each such input sequence corresponds to the requests allotted to a given facility. That is $n = \sum_j a_j$, where a_j is the number of requests allotted to the j-th facility. Also, we know that $a_j \leq k^*$ for all j. Additionally had the input been just the a_j requests, $\frac{C_1}{C_{OPT}}$ would have been less than $\mu(k^*)$. Now that the various clusters are given as input together, the cost paid by OPT is exactly the sum of the costs of the clusters separately. However for the online algorithm RFL the cost paid on the entire input might be more than the sum of the costs over the clusters (assuming that the input was just the cluster). Our initial guess was that this cost C_1 over the entire sequence might still be less than some constant times the sum of the costs of the clusters individually. However we have been unable to prove this.

We observe that the denominator has OPT's cost in the model with congestion. So OPT would have at most k^* requests allotted to any facility (Using Claim B.2). Let us now concentrate on any facility opened by OPT and look at the requests that are allocated to the facility. So let us write $S = S_1 \cup S_2 \cup ... \cup S_k$, where S is the entire collection of input requests and S_i is the set of input requests that were allotted to a particular facility and k is the total number of facilities opened by OPT. Let C_{S_i} represent the cost paid by OPT in the model with congestion if the input was just S_i instead of S and let $\tilde{C}_{S_i}$ represent the cost paid by OPT if the input was S_i in the model without congestion. Similarly let $C_{1_{S_i}}$ be the cost paid by RFL (recall that for this we are only looking at the model without congestion) over the requests in S_i but when the input is the entire input S.

$$\frac{C_1}{C_{OPT}} = \frac{C_1}{\sum_i C_{S_i}} \leq \frac{C_1}{\sum_i \tilde{C}_{S_i}} = \frac{\sum_i C_{1_{S_i}}}{\sum_i \tilde{C}_{S_i}} \leq \max_i \left\{ \frac{C_{1_{S_i}}}{\tilde{C}_{S_i}} \right\}. \tag{2}$$

Here the denominator, $\tilde{C}_{S_i}$ is the cost paid by OPT if the input was S_i in the model without congestion. However the numerator is $C_{1_{S_i}}$, which is the cost paid by RFL in the model without congestion when the input is the entire input S. Had the numerator been the cost paid by RFL in the model without congestion when the input is just S_i, we would have stated that this is exactly $\mu(|S_i|)$. Also since $|S_i| \leq k^*$ for all i, we would have obtained $\frac{C_1}{C_{OPT}} \leq \mu(k^*) = \mathcal{O}\left(\frac{\log k^*}{\log \log k^*}\right)$ for RFL.

The only problem is that $C_{1_{S_i}}$ is the cost paid by RFL, in the model without congestion when the input is the entire input S. This means that RFL might have extra facilities already open when the requests of S_i start coming in. Also new facilities might open between any two requests of S_i since other requests from S might be interleaved in between requests from S_i. The analysis of RFL in [9] actually goes through if there were open facilities available before the first request comes and also if new facilities appeared in between two requests (Interestingly, any algorithm that satisfies this property, could have been used as a black box instead of RFL). For the sake of completion, we add a thorough analysis of this in the full version of the paper [5]. This completes the proof that the competitive ratio of $MRFL$ is $\frac{\log k^*}{\log \log k^*}$. ◀

C Missing Proofs

Proof of Property 5.7. Let $X_i = \{(x_i, w_i), (x_{i+1} w_{i+1}), \ldots, (x_m, w_m)\}$.

Let C_i be the expected cost paid by $WRFL$ before a facility opens at any request in X_i, when the input is X_i. Note that now, we are interested in C_1, which is the expected cost paid by $WRFL$ before a facility opens in X. From Algorithm 1, we recall that $WRFL$ opens a new facility at the request x_i with probability p_i and with the remaining probability it assigns x_i to the nearest open facility.

One can see that $C_1 = p_1 \cdot 0 + (1 - p_1)(p_1 \cdot f + C_2)$ since with probability p_1 a facility is opened at x_1 itself and no cost is incurred before the first facility is opened but with probability $(1 - p_1)$, no facility is opened and a cost of $p_1 \cdot f = d(x_1, F) \cdot w_1$ is incurred. However since we have not yet opened any facility, in expectation C_2 additional cost will be incurred before a facility is opened - because of how C_2 is defined.

We can use the same argument for any C_i now. That is, when X_i is given as input, with probability p_i a facility is opened at x_i itself and no cost is incurred before the first facility is opened and with probability $(1 - p_i)$, no facility is opened and a cost of $p_i \cdot f$ is incurred

$$C_i = (1 - p_i)(p_i \cdot f + C_{i+1}).$$

Now notice that C_m is the expected cost paid by $WRFL$ before a facility opens when input is $\{(x_m, w_m)\}$. Therefore $C_m = p_m \cdot f \cdot (1 - p_m)$ since with probability $(1 - p_m)$, facility is not opened and then the cost paid is $p_m \cdot f$. This means that $C_m \leq f$.

Assuming $C_{i+1} \leq f$ for some i, $C_i \leq (1 - p_i)(p_i \cdot f + f) = f(1 - p_i)(1 + p_i) = f(1 - p_i^2) \leq f$.
$\therefore C_1 \leq f$. ◀

Proof of Property 5.8. Since the distance of x_i to the nearest open facility is $d(x_i, F)$, $WRFL$ would open a new facility with probability $\leq \frac{d(x_i, F) \cdot w_i}{f}$ and pay a cost f and with remaining probability (of at most 1) it would pay a cost of $d(x_i, F) \cdot w_i$. Therefore expected cost paid for the request x_i is at most

$$\frac{d(x_i, F) \cdot w_i}{f} \times f + 1 \times d(x_i, F) \cdot w_i = 2d(x_i, F) \cdot w_i \ . \quad ◀$$

An Optimal Algorithm for Sorting in Trees

Jishnu Roychoudhury
Princeton University, NJ, USA

Jatin Yadav
Indian Institute of Technology Delhi, India

Abstract

Sorting is a foundational problem in computer science that is typically employed on sequences or total orders. More recently, a more general form of sorting on partially ordered sets (or posets), where some pairs of elements are incomparable, has been studied. General poset sorting algorithms have a lower-bound query complexity of $\Omega(wn + n \log n)$, where w is the width of the poset.

We consider the problem of sorting in trees, a particular case of partial orders. This problem is equivalent to the problem of reconstructing a rooted directed tree from path queries. We parametrize the complexity with respect to d, the maximum degree of an element in the tree, as d is usually much smaller than w in trees. For example, in complete binary trees, $d = \Theta(1), w = \Theta(n)$. The previous known upper bounds are $O(dn \log^2 n)$ [19] and $O(d^2 n \log n)$ [1], and a recent paper proves a lower bound of $\Omega(dn \log_d n)$ [17] for any Las Vegas randomized algorithm. In this paper, we settle the complexity of the problem by presenting a randomized algorithm with worst-case expected $O(dn \log_d n)$ query and time complexity.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Sorting, Trees

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.7

Related Version *Previous Version*: `https://arxiv.org/abs/2205.15912`

1 Introduction

Sorting is a foundational problem in computer science that is typically employed on a set or multi-set of elements. Given an input set S of n elements, typical sorting problems determine the underlying total order or linear sequence of the elements. These algorithms assume that the elements are drawn from an ordered domain (such as the domain of integers). Most sorting algorithm use comparisons in which direct comparisons between pairs of elements allows the algorithms to acquire information about the total order [7]. Some recent works consider a restricted version of standard comparison sorting where only a subset of pairs are allowed to be compared [3, 4, 12, 15].

More recently, a more general form of sorting on partially ordered sets (or posets), where some pairs of elements are incomparable, has been studied [2, 5, 6, 8, 9]. Given a input set P of n elements, poset sorting algorithms determine the underlying partial order of the elements through *queries* to an oracle. The oracle's response to a query involving elements x and y is either the relation between x and y or a statement of their incomparability.

Sorting problems are inherently more challenging for partial orders compared to total orders. Sorting algorithms are generally evaluated by two complexity measures: the *query complexity*, the number of comparisons involved, and *total complexity*, the total number of computational operations involved. As each individual query can potentially be quite expensive (for example, requiring physical experiments), query complexity is equally important as total complexity. Faigle and Turán were the first to consider the problem of sorting partial orders [9], providing an algorithm with $O(wn \log n)$ query complexity, where n is the number of elements and w is the width of the poset. Daskalakis et al. [8] improved on this result

© Jishnu Roychoudhury and Jatin Yadav;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 7; pp. 7:1–7:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

to present an algorithm with optimal query complexity $O(wn + n \log n)$, and provided a separate algorithm with time complexity $O(w^2 n \log \frac{n}{w})$. A recent work in 2023 by Shaofeng H.-C. Jiang et al. [14] studies a generalized version of the poset sorting problem where a query graph G, defined on the same vertex set as the poset, is provided, and queries can only be made between pairs of elements which correspond to edges in the query graph.

Our work considers sorting tree posets, a particular case of partial orders. In a tree, the value of w is typically very high, leading to existing poset sorting algorithms performing poorly. Therefore, it is important to design efficient customized algorithms for sorting in tree posets with lower query and total complexity. We instead consider the parameter d, the maximum degree of a node in the natural arborescence representation of the tree poset. In contrast to width w, the degree d is typically much smaller; for example, $d = \Theta(1), w = \Theta(n)$ in complete binary trees. As with the poset sorting algorithms, our objective is to determine the underlying structure of the tree.

Apart from the theoretical significance of the problem, there are many practical applications: for example, constructing the evolutionary tree of coronavirus strains or general phylogenetic trees, or network mapping, among others. In addition, the final sorted tree can be used as input for tree searching algorithms, such as [10, 16]. This may be considered a tree analogue to the use of sorting to facilitate binary search in total orders. Onak and Parys [16] considered extending the concept of binary search to trees, whereas Heeringa, Iordan, and Lewis [10] considered searching in *dynamic* trees.

The problem we study is equivalent to the problem of reconstructing a rooted directed tree (arborescence) with **reachability** or **path queries**, as a comparison between two elements x and y of a tree poset can easily be simulated by asking two reachability queries (x, y) and (y, x) in the underlying rooted directed tree structure and vice-versa. This problem has received previous literature attention and there are currently two upper bounds with differing dependency on n and d. Wang and Honorio [19] provide a randomized algorithm with query complexity $O(dn \log^2 n)$, while Afshar et al. [1] provide a randomized algorithm with query complexity $O(d^2 n \log n)$.

Jagadish and Sen [13] considered the problem of reconstructing undirected trees using **separator queries**: given three vertices x, y, z, does y lie on the path between x and z? They give a deterministic algorithm with $O(dn^{1.5} \log n)$ query and time complexity. It should be noted that path queries are weaker than separator queries because there is a directed path from x to y in a rooted directed tree (or equivalently x is an **ancestor** of y) if and only if x lies on the path between the root and y in the underlying undirected tree structure. Nevertheless, their algorithm can be modified to work with path queries as well.

In a recent work, Bastide and Groenland [17] consider the problem of reconstructing connected undirected graphs with **distance queries**. Here, given two nodes (x, y), a query $\ell(x, y)$ returns the number of edges on the shortest path between them. They give a simple deterministic algorithm in the case of trees with a query complexity of $O(dn \log_d n)$. They also give a matching lower bound of $\Omega(dn \log_d n)$ queries for any randomized algorithm. This lower bound directly applies to our problem as well, as distance queries are stronger than comparision queries: for a rooted directed tree with root r, one can find if x is an ancestor of y by asking three distance queries $\ell(x, y)$, $\ell(r, x)$ and $\ell(r, y)$ in the undirected version of the tree. Thus, the fact that one needs $\Omega(dn \log_d n)$ comparision queries to sort a tree is stated as a corollary in their work (Corollary 4.10). Nevertheless, for the sake of simplicity and completeness, we borrow ideas from [17] to provide a shorter and more direct proof of the lower bound for **path queries**.

Many other previous works (e.g., [11, 18, 20, 21] have also considered the problem of reconstructing directed trees from distance queries. However, these works are limited by the specificity of distance as a query object. For instance, in evolutionary trees, it is far more natural to consider an ancestor relationship than a distance one. Our work provides an optimal algorithm for this most natural query object in trees.

2 Our contribution

We provide an optimal randomized algorithm for sorting tree posets using comparision queries, and equivalently reconstructing rooted directed trees using path queries:

▶ **Theorem 1.** *There exists a randomized algorithm which sorts a tree of n elements and maximum degree d in $O(dn \log_d n)$ expected worst-case query and time complexity.*

Table 1 The known bounds on the query complexity of randomized algorithms.

Work	Lower bound	Upper bound
[19]	$\Omega(n \log n)$	$O(dn \log^2 n)$
[1]	$\Omega(dn + n \log n)$	$O(d^2 n \log n)$
[17]	$\Omega(dn \log_d n)$	–
This work	–	$O(dn \log_d n)$

Both of the previous approaches ([1, 19]) for the upper bound rely on finding an **even-separator** edge, an edge such that the two subtrees remaining after removing the edge both have size at least $\frac{n}{d} - 1$, where n is the number of nodes in the original tree.

In contrast, our approach relies on finding a vertex such that the subtrees around the vertex each have size at most three-quarters that of the original tree. The actual details of finding such a vertex and then analysing the complexity are quite non-trivial. Another key difference is that in both [19] and [1], a bound on degree d is assumed to be known to the algorithm from the outset. In our problem formulation, the maximum degree d is not known and is instead discovered by the algorithm itself. This flexibility significantly improves the applicability of the solution. We only use d as a parameter in the complexity analysis.

3 Preliminaries

To precisely define the notions considered in this paper, we provide some formal definitions, many of which are set-theoretic analogues of graph-theoretic concepts. A partially ordered set, or poset, is a set of elements P together with a binary relation $\succ \subset P \times P$ which is irreflexive, transitive, and asymmetric. For a pair $(a, b) \in \succ$, we write $a \succ b$ and state that a dominates b, or b is dominated by a. If neither $a \succ b$ nor $b \succ a$, we say that a and b are *incomparable* and write $a||b$. A *maximal element*, or *root*, is an element x for which the set $\{y \in P : y \succ x\}$ is empty. Let the set of maximal elements of a poset $\mathcal{P}$ be $M(\mathcal{P})$. An *antichain* $A \subseteq P$ is a subset of mutually incomparable elements. The *width* $w(\mathcal{P})$ of poset $\mathcal{P}$ is the maximum cardinality of an antichain of the poset.

A *tree* is a poset $\tau = (T, \succ)$ where for all elements $t \in T$, the set of ancestors of t in the tree, $Anc_t = \{s \in T : s \succ t\}$, is well-ordered by the relation $\succ$. We will also sometimes refer to the elements of the tree as the *nodes* of the tree. A *child* of an element x is an element y such that $x \succ y$ and the set $\{z \in T : x \succ z \succ y\}$ is empty. For an element $t \in T$, we define ch_t to be the set of children of t. The *parent* of an element x, par_x, is an element y such that

$y \succ x$ and the set $\{z \in T : y \succ z \succ x\}$ is empty. Moreover, we define an *edge* as any pair (a, b) such that $b \in ch_a$, and the *degree* of an element t, deg_t, as the number of edges incident to t, i.e., $deg_t = |ch_t|$ for a maximal element, and $deg_t = |ch_t| + 1$ for all other elements. The *maximum degree* of the tree is $d(\tau) = \max(|M_\tau|, \max_{t \in T} deg_t)$. Note that $d(\tau)$ is dependent on the degree of elements in the tree and the number of maximal elements. The *subtree* of an element t is the set $\{t\} \cup \{s \in T : t \succ s\}$.

Trees in the set-theoretic sense may have multiple maximal elements (corresponding to graph-theoretic *forests*). However, we can easily modify any arbitrary tree to have only one maximal element. Suppose the elements of the tree are $\{t_1, t_2, \ldots, t_n\}$. We introduce a new element t_{n+1}, such that $t_{n+1} \succ t_i$ for all $i \in \{1, \ldots, n\}$, and sort the modified tree. In addition, we never need to make a query involving t_{n+1}, as it is known that t_{n+1} dominates all other elements. Thus, we will only consider trees with one maximal element for the rest of this paper.

Our algorithm finds the *edges* of the tree $\tau = (T, \succ)$, which is sufficient to sort the tree. Note that, after finding the edges, a standard $O(n)$ depth-first search can be run from the root, producing a data structure that stores the DFS pre-visit and post-visit times for all the vertices. This data structure that can respond to comparison queries in $O(1)$, as a vertex x is an ancestor of vertex y (or equivalently $x \succ y$) if and only if $\text{pre}[y] \in (\text{pre}[x], \text{post}[x]]$, thus fully mimicking the oracle.

4 Algorithm Outline

Our main algorithm, `GET-EDGES` (detailed pseudocode can be found in Appendix B), relies on finding a *good* separating element in the tree and then dividing the tree into disjoint sub-problems around this element, such that no sub-problem has a large size. An obvious first choice for such a node is the centroid of the tree:

▶ **Definition 2** (Centroid). *The centroid of a tree T with n nodes is a node c, such that the size of the subtree of c is $\geq \frac{n}{2}$, and for any of its children, the subtree size is $< \frac{n}{2}$.*

It is well known and easy to see that there exists a unique centroid for any tree. Nevertheless, we include the proof in Appendix A (Lemma 8) for the sake of completeness.

To find the centroid c with probability at least $\frac{1}{2}$, we can select a random node x, and find its set of ancestors. This set forms a total order, and can be sorted using a standard comparision-based sorting algorithm. Now, we binary search on this list and return the lowest ancestor with subtree size $\geq \frac{n}{2}$. If x lies in the subtree of c (which occurs with probability at least $\frac{1}{2}$, as the subtree size of c is $\geq \frac{n}{2}$), this process will return c. Since sorting takes $O(n \log n)$ time and $O(\log n)$ subtree size computations during the binary search can be done in $O(n)$ each by iterating over all the nodes, this takes $O(n \log n)$ time.

To save a factor of $\log n$, we will instead use a *pseudo-centroid* as the separating element:

▶ **Definition 3** (Pseudo-Centroid). *A pseudo-centroid of a tree T with n nodes is a node c, such that the size of the subtree of c is $\geq \frac{n}{4}$, and for any of its children, the subtree size is $< \frac{n}{2}$. In particular, note that the centroid is also a pseudo-centroid.*

With this definition in mind, `GET-EDGES` now proceeds in three steps:

1. Find a pseudo-centroid. To accomplish this, we will later describe a method (similar to the above described method for finding a probable centroid) to find a probable (with probability $\geq \frac{1}{8}$) pseudo-centroid c in $O(n)$ expected time. Then, we test if c is a pseudo-centroid, and if it is not, we repeat the whole process.

2. Divide the elements of the tree (except c) into sub-problems around the element c. The subtree of each child of c forms a subproblem and all the elements of the tree except the subtree of c form an additional subproblem. We refer to the former as *downward subproblems* and the latter as the *upward subproblem* for brevity.
3. Solve each subproblem recursively. Each subproblem is a tree which constitutes a (weakly) connected subgraph of the original graph. By the definition of pseudo-centroids, each sub-problem will have $\leq \frac{3n}{4}$ nodes.

We will design the algorithm to test whether a node c is a pseudo-centroid in such a way that:

1. If c is not a pseudo-centroid, the test returns a negative result in $O(n)$ expected time.
2. If c is a pseudo-centroid and divides the tree into subproblems $x_1, x_2, \ldots, x_k$ with $|x_1| \geq \cdots \geq |x_k|$, then the algorithm results a positive result, along with all the k subproblems, in $O(n + \sum_{i=1}^{k} i|x_i|)$ expected time.

Then, the recursion for the expected time complexity becomes

$$T_d(n) = O(n) + \sum_{i=1}^{k} (O(i|x_i|) + T_d(|x_i|))$$

where $k \leq d$, $\sum_{i=1}^{k} |x_i| = n - 1$ and $1 \leq |x_i| \leq \frac{3n}{4}$ $\forall\, 1 \leq i \leq k$. We will prove via a charging argument on the *recursion tree* that $T_d(n) = O(dn \log_d n)$.

5 Algorithm Details and Complexity Analysis

We first describe our algorithm `GET-PROBABLE-PSEUDO-CENTROID` for finding a probable (with probability $\geq \frac{1}{8}$) pseudo-centroid in $O(n)$ expected time. We assume sufficiently large n (greater than some constant) in the discussion to follow to avoid minor details such as the possibilities of $\log n = 0$ or $\frac{1}{\log n} > 1$, and for some inequalities to be true.

Define the subtree size of a node i as S_i. Also, let $p = \frac{1}{\log n}$. Now,

1. Randomly sample a subset V of nodes by adding each node to V with an independent probability of p. Thus, $\mathbb{E}(|V|) = \frac{n}{\log n}$. We define the sampled subtree size, F'_i to be the number of elements in V that lie inside the subtree of i. Clearly, $\mathbb{E}(F'_i) = S_i p$. Also, we define the estimated subtree size of i to be $S'_i = \frac{F'_i}{p}$, so that $\mathbb{E}(S'_i) = S_i$.
2. Select a random node x. We will output the lowest ancestor y of x with $S'_y \geq \frac{n}{2} - \frac{1}{p}$ as our (probable) pseudo-centroid. Let Y be the set of ancestors of x, along with x itself. Note that, for any ancestor u of v, $S'_u \geq S'_v$. One way to find this lowest ancestor y is to first sort Y and then binary search on Y. But sorting Y can take $n \log n$ queries. To fix this, we sample a subset Z of Y by adding each node in Y to Z with an independent probability of p. Thus, $\mathbb{E}(|Z|) = \frac{\mathbb{E}(|Y|)}{\log n}$.
3. Sort Z according to the order relation $\succ$. Then, use a modified binary search on Y to find the pseudo-centroid. This modified binary search runs in two phases:
   a. Binary search on Z to find an *estimated* pseudo-centroid within Z.
   b. Find the position of the actual pseudo-centroid from the expected $O(\log n)$ elements of Y that lie between the estimated pseudo-centroid and the immediate next element in Z.

Note that binary searching is valid because for any ancestor u of v, $S'_u \geq S'_v$. We now prove bounds on the complexity of this algorithm and its probability of success.

Algorithm 1 `GET-PROBABLE-PSEUDO-CENTROID`.

Input : A set T of nodes that form a tree $\tau = (T, \succ)$ with $\succ$ unknown, and an oracle that answers for any pair of nodes $a \in T, b \in T$ whether $a \succ b$.
Output : A probable pseudo centroid of T

```
n ← |T|
p ← 1/log n
V ← {}
for i ∈ T do
    Add i to V with probability p
end
Choose a random element x from T.
Y ← {x} ∪ {y ∈ T | y ≻ x}
Z ← {GET-ROOT(T)}
for i ∈ Y do
    Add i to Z with probability p
end
Sort Z according to the order relation ≻
Let Z = {z_1, z_2, ... z_k} where z_1 ≻ z_2 ≻ ... z_{k-1} ≻ z_k
Define S'_j to be the estimated size of subtree of j, S'_j = |({j} ∪ {i ∈ T | j ≻ i}) ∩ V| / p
t ← z_1
if S'_{z_1} ≥ n/2 − 1/p then
    Binary search to find the largest i, such that S'_{z_i} ≥ n/2 − 1/p
    L ← {z_i}
    for u ∈ Y do
        if (i ≠ k and z_i ≻ u and u ≻ z_{i+1}) or (i = k and z_i ≻ u) then
            L ← L ∪ {u}
        end
    end
    Sort L according to the order relation ≻
    Let L = {l_1, l_2, ..., l_r}, where l_1 ≻ l_2 ... l_{r-1} ≻ l_r
    Binary search to find the largest i, such that S'_{l_i} ≥ n/2 − 1/p
    t ← l_i
end
return t
```

▶ **Lemma 4.** *The algorithm* `GET-PROBABLE-PSEUDO-CENTROID` *finds a pseudo-centroid with probability* $\geq \frac{1}{8}$ *in* $O(n)$ *expected time.*

Proof. It is simple to verify that the algorithm takes $O(n)$ expected time by considering each individual step. Specifically:

1. Sampling V takes $O(n)$ time.
2. Selecting x, computing Y, and sampling Z takes $O(n)$ time.
3. Sorting Z takes $O(|Z| \log |Z|) = O(|Z| \log n)$ time. Since $\mathbb{E}(|Z|) = \frac{|Y|}{\log n} \leq \frac{n}{\log n}$, this takes $O(n)$ expected time. The modified binary search also takes $O(n)$ time, as shown below:
   a. Binary searching Z takes $O(\log n)$ rounds. In each round, we compute the estimated subtree size for some element i, S'_i, which takes $O(\frac{n}{\log n})$ time since $\mathbb{E}(|V|) = \frac{n}{\log n}$. Thus, this step takes $O(n)$ expected time total.

b. Finding the list L of elements that lie in-between takes $O(n)$ time, as it requires only two comparison queries per element in Y. Moreover, since $\mathbb{E}(L) = O(\log n)$, sorting L takes expected $\mathbb{E}(|L| \log |L|) = O(\mathbb{E}(|L| \log n)) = O(\log^2 n) = O(n)$ time, and binary searching L takes $O(\log n)$ rounds. In each round, we compute an estimated subtree size in expected $O(\frac{n}{\log n})$, and thus the total expected time complexity is $O(n)$.

We thus concentrate on the probability of success. Let the centroid be c (this is the unique centroid, not just a pseudo-centroid). We first prove that, for sufficiently large n, if all of the following three events occur, we get a pseudo-centroid.

1. $x = c$ or $c \succ x$
2. $S'_c \geq \frac{n}{2} - \frac{1}{p}$
3. For all $i \in T$, $S'_i \leq S_i + \frac{n}{10}$

The algorithm returns the minimal ancestor y of x with $S'_y \geq \frac{n}{2} - \frac{1}{p}$ (let it be c'). If $c' = c$, we are done. Else, events 1 and 2 imply $c \succ c'$. Now, $S'_{c'} \geq \frac{n}{2} - \frac{1}{p}$ and $S_{c'} \geq S'_{c'} - \frac{n}{10}$ (event 3). Adding these two gives $S_{c'} \geq \frac{n}{2} - \frac{1}{p} - \frac{n}{10} = \frac{2n}{5} - \log n \geq \frac{n}{4}$ for sufficiently large n. Also, $S_{c'} < \frac{n}{2}$ as c' lies strictly inside the subtree of c, and all children of c have sizes $< \frac{n}{2}$. So, for every child r of c', $S_r < S_{c'} < \frac{n}{2}$. Thus, c' is a pseudo-centroid.

We now proceed to proving that the probability of these 3 events all occurring simultaneously is $\geq \frac{1}{8}$ for sufficiently large n. Let us try to get an upper bound on at least one of these events not occurring. Let p_1, p_2, p_3 be the probabilities of events $1, 2, 3$ **not** occurring respectively. We will establish bounds on all three of them. Firstly, $\mathbf{p_1} \leq \mathbf{\frac{1}{2}}$ as at least half of the nodes lie inside the subtree of c.

To get an upper bound on p_2, note that F'_c follows a binomial distribution with parameters $S_c \geq \frac{n}{2}$ and $p = \frac{1}{\log n}$. As any median of a binomial distribution with parameters n and p is $\geq \lfloor np \rfloor$, we have $S'_c = F'_c/p \geq \lfloor \frac{pn}{2} \rfloor / p \geq \frac{n}{2} - \frac{1}{p}$ with probability at least $\frac{1}{2}$. Therefore, $\mathbf{p_2} \leq \mathbf{\frac{1}{2}}$.

Finally, we will prove that $\mathbf{p_3} \leq \mathbf{\frac{1}{4}}$ for sufficiently large n. Indeed, consider an element i, with k elements in its subtree, $u_1, u_2, \ldots u_k$, where $k = S_i$. Let X_j be an indicator variable for whether u_j was chosen in V. All X_j's are independent and identically distributed with $\mathbb{E}(X_j) = p$. Also, $F'_i = X_1 + X_2 + \ldots X_k$, and $\mathbb{E}(F'_i) = kp$. Since we want a total error of $\leq \frac{np}{10}$ in F'_i, we use the Chernoff bound with $\mu = kp, \delta = \frac{n}{10k} \geq \frac{1}{10}$ to get:

$$\begin{aligned}
\mathbb{P}\left(S'_i > S_i + \frac{n}{10}\right) &= \mathbb{P}\left(F'_i > kp + \frac{np}{10}\right) \\
&= \mathbb{P}\left(F'_i > \mu(1+\delta)\right) \\
&\leq \exp\left(-\frac{\mu}{3}\delta \min(1, \delta)\right) \\
&\leq \exp\left(-\frac{kp}{3}\frac{n}{10k}\frac{1}{10}\right) \\
&= \exp\left(\frac{-np}{300}\right) \qquad (1)
\end{aligned}$$

Using the union bound, $p_3 \leq n \cdot \exp\left(\frac{-np}{300}\right) = n \exp\left(\frac{-n}{300 \log n}\right) \leq \frac{1}{4}$ for sufficiently large n.

Now, let us suppose we fail if at least one of the events $1, 2, 3$ does not occur. The probability of failure is the sum of the probability of event 1 not occurring and the probability of event 1 occurring but at least one of the events 2 and 3 not occurring. Noticing that event 1 is independent of both events 2 and 3, and applying the union bound, we get that the probability of failure is at most:

$$p_1 + (1 - p_1)(p_2 + p_3) \leq p_1 + \frac{3}{4}(1 - p_1) = \frac{3}{4} + \frac{p_1}{4} \leq \frac{7}{8}$$

Therefore, we find a pseudo-centroid with probability at least $\frac{1}{8}$. ◀

Algorithm 2 `TEST-PSEUDO-CENTROID`.

Input : A set T of nodes that form a tree $\tau = (T, \succ)$ with $\succ$ unknown, and an oracle that answers for any pair of nodes $a \in T, b \in T$ whether $a \succ b$. Also a candidate pseudo-centroid c

Output : A pair (b, S), where b is a boolean denoting if c is a pseudo-centroid. If b is `true`, S is the set of subproblems when T is divided into around c.

```
n ← |T|
H ← T \ c
S ← []
R ← {x ∈ H : x ≻ c}
if |R| > 3n/4 then
    return (false,[])
end
Append R to S
H ← H \ R
while H ≠ ∅ do
    Let H = h_1, h_2, ..., h_m
    Choose a random element x from H
    for i = 1, ..., m do
        if h_i ≻ x then
            x ← h_i
        end
    end
    X ← {x} ∪ {y ∈ H : x ≻ y}
    if |X| ≥ n/2 then
        return (false,[])
    end
    Append X to S
end
return (true,S)
```

We now describe our algorithm `TEST-PSEUDO-CENTROID`, which simultaneously tests if an element c is a pseudo-centroid and divides the tree into subproblems around c if it is. The algorithm first finds the upward subproblem in $O(n)$, removes the elements from the list of unaccounted elements, and tests if the subtree of c has size at least $\frac{n}{4}$, returning `false` if it does not. After this, we repeatedly find one of the downward subproblems, check if the subproblem has size less than $\frac{n}{2}$ (returning `false` if it does not), and remove the subproblem elements from the list of unaccounted elements until there are no unaccounted elements remaining, at which point we return the list of subproblems. We can find a downward subproblem in $O(|H|)$, where H is the set of unaccounted elements, by doing the following. First, set x to be a random element from H. Then, iterate through all elements in H. If h_i is the current element, set $x \leftarrow h_i$ if $h_i \succ x$. Thus, the final value of x will be that of one of the children of c. Then, we find all elements in the subtree of x by comparing each element in H against x, and thus find a subproblem in $O(|H|)$ time.

The algorithm correctness is obvious and as such we prove its complexity.

▶ **Lemma 5.** *Suppose c divides the tree into k subproblems $x_1, \ldots, x_k$, such that $|x_1| \geq \cdots \geq |x_k|$. Then:*

- *If c is not a pseudo-centroid, `TEST-PSEUDO-CENTROID` runs in $O(n)$ expected time.*
- *If c is a pseudo-centroid, `TEST-PSEUDO-CENTROID` runs in $O(n + \sum_{i=1}^{k} i|x_i|)$ expected time.*

Proof. We first prove that `TEST-PSEUDO-CENTROID` runs in $O(n)$ expected time when c is not a pseudo-centroid. There are two cases. The first is when the size of the subtree of c is less than $\frac{n}{4}$. This is detected immediately after finding the upward subproblem, which takes only $O(n)$ time. The second case is that one of the downward subproblems x_l has size at least $\frac{n}{2}$. In this case, note that we have a probability of at least $\frac{1}{2}$ of finding x_l in each iteration, as the subproblem found contains the random element chosen at the beginning of the iteration. Thus, we are expected to detect failure in a constant number of iterations. Since each iteration runs in $O(n)$, the expected complexity is $O(n)$.

We now prove that `TEST-PSEUDO-CENTROID` runs in $O(n + \sum_{i=1}^{k} i|x_i|)$ expected time when c is a pseudo-centroid. First, $O(n)$ time is used to find the upward subproblem.

Let us analyse the expected number of comparisions between two downward subproblems x_i and x_j with $i \neq j$. The probability that x_i is found before x_j is $\frac{|x_i|}{|x_i|+|x_j|}$. When this happens, each element of subproblem x_j is compared against twice, once when finding the maximal element of x_i and once when finding the list of elements of x_i. Similarly, we can account for the expected comparisions when x_j is found before x_i, and the total expected number of comparisions between x_i and x_j is therefore $2|x_j|\frac{|x_i|}{|x_i|+|x_j|} + 2|x_i|\frac{|x_j|}{|x_i|+|x_j|} = \frac{4|x_i||x_j|}{|x_i|+|x_j|}$

The cost of finding the upward subproblem x_p has already been accounted for. Regardless, we also charge a cost of $4\frac{|x_p||x_j|}{|x_p|+|x_j|}$ for the number of comparisons between x_p and x_j for simplicity, which is acceptable since we are proving an upper bound. In addition, there are some comparisons made between two elements of the same subproblem x_i. However, there are only $O(|x_i|)$ such comparisons, and thus the total over all subproblems is $O(n)$.

We observe that for any $1 \leq j < i \leq k$, $\frac{|x_i||x_j|}{|x_i|+|x_j|} \leq \min(|x_i|, |x_j|) = |x_i|$. The total number of comparisions is therefore bounded by:

$$O(n) + \sum_{1 \leq j < i \leq k} O(|x_i|) = O(n) + \sum_{i=1}^{k} O(i|x_i|)$$

It is easy to see that in each iteration both the runtime and the number of comparisions are $\Theta(|H|)$, so the runtime also follows the same bound of $O(n) + \sum O(i|x_i|)$. ◀

We finally prove the main result of the paper by showing that `GET-EDGES` runs in expected $O(dn \log_d n)$ time.

▶ **Theorem 6.** *`GET-EDGES` sorts a tree of n elements and maximum degree d in $O(dn \log_d n)$ expected worst-case query and time complexity.*

Proof. Due to Lemmas 4 and 5, the recurrence relation for the time complexity of `GET-EDGES` is

$$T_d(n) = O(n) + \sum_{i=1}^{k} (O(i|x_i|) + T_d(|x_i|))$$

where $x_1, \ldots, x_k$ are the subproblems divided around the pseudo-centroid, and $|x_1| \geq \cdots \geq |x_k|$.

We prove the complexity of this recurrence relation by considering the *recursion tree* of the algorithm. The root of this recursion tree is the entire tree, and for any node, its children are the subproblems divided around the pseudo-centroid. Note that this recursion tree has exactly n nodes.

We immediately note that the sum of all $O(n)$ terms is $O(n \log n)$, as the recursion tree has $O(\log n)$ levels (since child subproblems have size at most $\frac{3}{4}$ that of a parent subproblem), and the sum of subproblem sizes at each level is at most n.

We now consider the sum of all $\sum_{i=1}^{k} O(i|x_i|)$ terms. For each node in the recursion tree, we rank its children $x_1, \ldots, x_k$ in non-increasing order of subproblem size, such that $|x_1| \geq \cdots \geq |x_k|$. Then, the cost of a child node x_i is $O(i|x_i|)$. Note that since this subproblem has size $|x_i|$, its subtree in the recursion tree also has $|x_i|$ nodes, including itself. We can now distribute the cost of $O(i|x_i|)$ by charging a cost of i to each node in the subtree.

Let v_z be the total cost charged to a node z. We now prove $v_z = O(d \log_d n)$ for all z. Consider the path from node z to the root of the recursion tree. For each node b in the path with parent a, if b had charged a cost of i to z, then at least i children of a have size at least $|s_b|$, where $|s_b|$ is the subtree size of b. However, we note then that the subtree size of a is at least $i|s_b|$. Moreover, the root has subtree size n.

Specifically, let a be an ancestor of node z, and let b be the unique node which is a child of a and an ancestor of z (or z itself). If we let $f_{z,a}$ be the rank of b within the children of a (according to the ranking scheme defined above), then we have:

$$v_z = \sum_{a \in A_z} f_{z,a}$$

where A_z is the set of ancestors of node z in the recursion tree. We also know that $\prod_{a \in Anc_z} f_{z,a} \leq n$, since if there are $f_{z,a}$ nodes with subtree size at least $|s_b|$, the subtree size of a is at least $f_{z,a}|s_b|$. In addition, we know that $f_{z,a} \leq d$ for all (z, a) (as each node in the recursion tree has at most d children), and that z has at most $O(\log n)$ ancestors. We use this information to show in Lemma 9 in the Appendix that $v_z = O(d \log_d n)$.

Since the sum of all $O(n)$ terms is $O(n \log n)$, and the sum of all $\sum_{i=1}^{k} O(i|x_i|)$ terms is $O(dn \log_d n)$, the total complexity is $O(n \log n + dn \log_d n) = O(dn \log_d n)$, as required. ◀

Note that the analysis of our algorithm is tight for complete $d-$ary trees. In the next section, we prove a matching lower bound considering the case of *almost* complete $d-$ary trees.

6 Lower bound

A lower bound of $\Omega(dn \log_d n)$ **distance queries** to reconstruct a tree of maximum degree d is proved in [17], and as a corollary this bound is valid for our case (**path queries**) as well, as each path query can be answered using 3 distance queries. Here, borrowing ideas from [17], we give a simpler and more direct proof of this lower bound in the case of path queries.

Assume $d \geq 3$. Consider a tree with $k+1$ levels, where for each $i = 0, 1, \ldots k$, the ith level is defined as the set of nodes at a distance i from the root. All nodes on the level k are leaves. Each node at a level $0 \leq i < k-1$ has exactly d children, and each node on the level $k-1$ has exactly 1 child. In other words, the first k levels form a complete $d-$ary tree, where for each $i = 0, 1, \ldots k-1$, the ith level contains d^i nodes, and there are d^{k-1} nodes on level k. The number of nodes in the tree is therefore $n = d^0 + d^1 + \ldots d^{k-1} + d^{k-1}$. Suppose the value of d and the whole subgraph consisting of the first k levels $0, 1, \ldots k-1$ are provided for free by the oracle, without any queries. Let the set of the nodes at level $k-1$ be X and

the set of the nodes at level k be Y. Let $m = |X| = |Y| = d^{k-1}$ be the number of nodes on the last level. Note that $m \geq \frac{n}{3}$. Let's call such a tree a *balanced tree* and let $\mathcal{B}$ be the set of all balanced trees. Note that the number of balanced trees is $B = |\mathcal{B}| = m!$, as we need to choose a distinct parent in X for each node in Y.

We want to apply Yao's principle to prove a lower bound on the number of queries required to reconstruct a *balanced tree*. Ideally, we would want to be able to consider each node in Y independently and prove a lower bound on the number of queries required to find its parent. But the fact that each node in X has exactly 1 child means that the parents of the nodes in Y have some correlation. So, let's first relax this condition, even though it would mean that the maximum degree can now exceed d. Let's define the set of *relaxed trees*, $\mathcal{T}$ to be the set of all the trees where the first k levels are as provided, and the parent of any node in Y can be any node in X. Note that $|\mathcal{T}| = m^m$, as for each node in Y, there are m choices for its parent.

First, we prove that if the input tree was chosen at random from $\mathcal{T}$, any deterministic algorithm must make at least $\Omega(mdk)$ queries with a very high probability. Clearly, any query that can give some useful information must be of this form: given a node y on the last level and a node x from the first k levels, is x an ancestor of y?

▶ **Lemma 7.** *Assume $d \geq 3$ and $k \geq 10$. Any deterministic algorithm that reconstructs a tree chosen at random from $\mathcal{T}$ uses at least $\frac{mdk}{8}$ queries with a **NO** answer, with probability $\geq 1 - e^{-m}$*

Proof. Choosing the tree at random from $\mathcal{T}$ is equivalent to choosing the parent of each of the m nodes in Y, independently and uniformly at random from X. Suppose we want to find the parent (say p) of some node $y \in Y$.

Suppose the algorithm knows a node z at some level $0 \leq i \leq k-1$, such that, given the set of queries asked in the past, p can be any of the nodes in the subtree of z at the level $k-1$, with equal probability. Initially, $i = 0$. When i equals $k-1$, we are done. Given z, let q_1 be the first vertex queried by the algorithm, and for each $u > 1$, define q_u to be the next queried vertex if the answer to q_{u-1} was **NO** (meaning that q_{u-1} is not an ancestor of y). Suppose that, out of the children $z_1, z_2, \ldots, z_d$ of z, z_r was the child which is an ancestor of y. The variable r is distributed uniformly among $[d]$. Let q_j be the first node in the sequence $q_1, q_2, \ldots$ that lies in the subtree of z_r (note that such a node must exist). Clearly, all of $q_1, q_2, \ldots q_{j-1}$ must be queried by the algorithm, all resulting in an answer of **NO**. When the algorithm queries q_j, instead of a **YES/NO** answer, the oracle replies that y lies in the subtree of z_r, and that the current query (q_j) doesn't count towards the total number of queries. This step only gives the algorithm extra information without any charge on the number of queries, and the algorithm can choose to ignore this information and query q_j again (we give a formal proof justifying this step in the Appendix, in Lemma 10). For each $t \in [d]$, let f_t be the smallest index h, such that q_h lies in the subtree of z_t. The number (say $Q_{y,i}$) of $t \in [d]$ with $f_t < f_r$ is a random integer in $\{0, 1, \ldots d-1\}$. Clearly at least $Q_{y,i}$ queries must be made before the algorithm knows the child z_r whose subtree contains y. The total number of queries asked therefore is at least a sum of $m \cdot (k-1)$ independent random variables each in the range $[0, d-1]$ with mean $\frac{d-1}{2}$. Applying Hoeffding's inequality yields that the probability that the number of queries asked is less than $m(d-1)(k-1)/4$ is at most $\exp\left(\frac{-2(m(k-1)(d-1)/4)^2}{m(k-1)(d-1)^2}\right) = e^{-(m(k-1))/8}$, and using $k \geq 10, d \geq 3$ yields the required result. ◀

This means that, given $d \geq 3, k \geq 10$, for any deterministic algorithm, there are at most $L = e^{-m}|\mathcal{T}| = e^{-m}m^m$ trees for which it takes less than $\frac{mdk}{8}$ queries. Also, the number of balanced trees is $B = m!$ Therefore, $L \leq \frac{B}{2}$ using $n! \geq \frac{n^{n^n}}{e^{n-1}}$ for all $n \geq 1$. We claim that, any deterministic algorithm which is able to correctly reconstruct all balanced trees must ask at least $\frac{mdk}{8}$ queries with a **NO** answer, for at least half of the balanced trees. Suppose this were not the case for a deterministic algorithm D. Consider an algorithm D' which is able to correctly reconstruct all trees in $\mathcal{T}$. D' first runs the algorithm D. If the original tree was balanced, the algorithm must already have the tree, and can verify it by querying for all the edges (note that if the tree was balanced, all these queries result in a **YES** answer). If the verification fails, D' just asks all the $\binom{n}{2}$ queries to reconstruct the tree from scratch. So, for at least $B/2 + 1 > L$ trees, D' makes $\leq \frac{mdk}{8}$ queries with a **NO** answer, a contradiction to Lemma 7.

It then follows from Yao's principle that the worst-case query complexity of any randomized algorithm must be at least $\frac{mdk}{16} \geq \frac{ndk}{48} = \Omega(nd \log_d(n))$. We note that this result holds even when $k < 10$ as there is a pre-existing bound of $\Omega(nd)$ which is only a constant off $\Omega(nd \log_d(n))$, since $n = O(d^{10})$ for $k < 10$.

7 Concluding Remarks

In this paper, we have considered the problem of sorting in trees, a particular case of partial orders. and presented an **optimal** randomized algorithm for sorting a tree poset in worst-case expected $O(dn \log_d n)$ query and time complexity, where d is the maximum degree of the tree. An interesting direction for future work is resolving the **deterministic** query complexity of tree sorting. Currently, to the best of our knowledge, the best known upper bound is $O(dn^{1.5} \log n)$ [13], and the best known lower bound is $\Omega(nd \log_d n)$.

References

1. Ramtin Afshar, Michael T. Goodrich, Pedro Matias, and Martha C. Osegueda. Reconstructing Biological and Digital Phylogenetic Trees in Parallel. 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 3:1–3:24, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2020.3.
2. Ramtin Afshar, Michael T Goodrich, Pedro Matias, and Martha C Osegueda. Parallel network mapping algorithms. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 410–413, 2021.
3. Noga Alon, Manuel Blum, Amos Fiat, Sampath Kannan, Moni Naor, and Rafail Ostrovsky. Matching nuts and bolts. In *SODA*, pages 690–696. Citeseer, 1994.
4. Indranil Banerjee and Dana Richards. Sorting under forbidden comparisons. In *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016)*. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.
5. Arindam Biswas, Varunkumar Jayapaul, and Venkatesh Raman. Improved bounds for poset sorting in the forbidden-comparison regime. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 50–59. Springer, 2017.
6. Jean Cardinal and Samuel Fiorini. On generalized comparison-based sorting problems. In *Space-Efficient Data Structures, Streams, and Algorithms*, pages 164–175. Springer, 2013.
7. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.
8. C Daskalakis, RM Karp, E Mossel, SJ Riesenfeld, and E Verbin. Sorting and selection in posets. *SIAM Journal on Computing*, 40(3), 2011.
9. U Faigle and Gy Turán. Sorting and recognition problems for ordered sets. *SIAM Journal on Computing*, 17(1), 1988.

10 Brent Heeringa, Marius Cătălin Iordan, and Louis Theran. Searching in dynamic tree-like partial orders. In *12th international conference on Algorithms and data structures (WADS)*, 2011.

11 Jotun J Hein. An optimal algorithm to reconstruct trees from additive distance data. *Bulletin of mathematical biology*, 51(5):597–603, 1989.

12 Zhiyi Huang, Sampath Kannan, and Sanjeev Khanna. Algorithms for the generalized sorting problem. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 738–747. IEEE, 2011.

13 M. Jagadish and Anindya Sen. Learning a bounded-degree tree using separator queries. In Sanjay Jain, Rémi Munos, Frank Stephan, and Thomas Zeugmann, editors, *Algorithmic Learning Theory - 24th International Conference, ALT 2013, Singapore, October 6-9, 2013. Proceedings*, volume 8139 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2013. doi:10.1007/978-3-642-40935-6_14.

14 Shaofeng H.-C. Jiang, Wenqian Wang, Yubo Zhang, and Yuhao Zhang. Algorithms for the generalized poset sorting. *arXiv preprint*, 2023. arXiv:2304.01623.

15 William Kuszmaul and Shyam Narayanan. Stochastic and worst-case generalized sorting revisited. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1056–1067. IEEE, 2022.

16 K. Onak and P. Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

17 Carla Groenland Paul Bastide. Optimal distance query reconstruction for graphs without long induced cycles. *arXiv preprint*, 2023. arXiv:2306.05979.

18 Lev Reyzin and Nikhil Srivastava. On the longest path algorithm for reconstructing trees from distance matrices. *Information processing letters*, 101(3):98–100, 2007.

19 Zhaosen Wang and Jean Honorio. Reconstructing a bounded-degree directed tree using path queries. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 506–513. IEEE, 2019.

20 Michael S Waterman, Temple F Smith, Mona Singh, and William A Beyer. Additive evolutionary trees. *Journal of theoretical Biology*, 64(2):199–213, 1977.

21 Li Zhang, Valerie King, and Yunhong Zhout. On the complexity of distance-based evolutionary tree reconstruction. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, page 444. SIAM, 2003.

A Omitted Proofs

▶ **Lemma 8.** *For any tree, the centroid exists and is unique.*

Proof. Consider the following algorithm. We start at the root, and keep recursing to the child with subtree size $\geq \frac{n}{2}$ until there is no such child. This maintains the invariant that any centroid must be inside the subtree of the current element. This process must terminate as the number of elements is finite, and it can only terminate on a centroid (let it be c). Any other centroid must lie in the subtree of c, but all children of c have subtree sizes less than $\frac{n}{2}$, so c must be the only centroid. ◀

▶ **Lemma 9.** *Given integers $1 \leq d \leq n$, and $k = O(\log n)$. For any k real numbers $x_1, x_2, \ldots x_k$, satisfying $\prod_{i=1}^{k} x_i \leq n$, and $1 \leq x_i \leq d$ for every i, $\sum_{i=1}^{k} x_i = O(d \log_d(n))$*

Proof. Consider any valid tuple $(x_1, x_2, \ldots x_k)$ which maximizes the sum. There can be at most $\log_d n$ values equal to d as the product is $\leq n$, and all values are ≥ 1. If there were two values p, q both in the range $(\sqrt{d}, d)$, we could have replaced them by $d, \frac{pq}{d}$, getting a strictly greater sum. So, there is at most one value in the range $(\sqrt{d}, d)$. Thus, the sum is $\leq k\sqrt{d} + d + d \log_d n = O(\sqrt{d} \log n) + d + d\frac{\log n}{\log d} = O(d \log_d n)$. ◀

Let $\mathcal{O}$ be an oracle that answers path queries with **YES** or **NO** (given vertices x, y, is x an ancestor of y in the rooted directed tree?). Say the root of the tree is already known. For a deterministic algorithm D, let $c(D, I)$ be the number of queries that D asks $\mathcal{O}$ to reconstruct a rooted directed tree I.

We will now consider a different oracle that can (a finite number of times) refuse to answer the path query and instead give some other information for free (without charging the current query to the cost). We will prove that any lower bound that can be proved for such an oracle can be extended to $\mathcal{O}$ as well. Formally, let $\mathcal{O}'$ be an oracle that, given two vertices x, y returns a pair (P, Q), where P is **YES/NO/IGNORE**, and Q is any node of the tree that is an ancestor of y. Call such a response to a query *charged* if P equals **YES/NO** and *free* otherwise. $\mathcal{O}'$ can give at most n^2 *free* responses, where n is the number of vertices in the tree. For a deterministic algorithm D', let $c'(D', I)$ be the number of **charged** responses given by $\mathcal{O}'$ while D' recovers I.

▶ **Lemma 10.** *For any deterministic algorithm D that queries $\mathcal{O}$, there exists a deterministic algorithm D' that queries $\mathcal{O}'$, such that for any tree I, $c'(D', I) = c(D, I)$.*

Proof. Given D, let us construct an algorithm D' such that $c(D, I) = c'(D', I)$ for all I. Consider a query made by D to O, with a response P, that is either **YES** or **NO**. D' simply keeps on making that same query to $\mathcal{O}'$ until it gets a charged response, say (P', Q'). Clearly P' must be equal to P, and D' can choose to ignore Q'. So, D' gets one charged response for every query made by D. ◀

This proof can also be extended to the case where the definitions of both $c(D, I)$ and $c'(D', I)$ are changed to the number of **NO** responses received by the algorithm.

B Algorithm Pseudocode

Algorithm 3 GET-EDGES.

```
Input  : A set T of nodes that form a tree τ = (T, ≻) with ≻ unknown, and an oracle that
         answers for any pair of nodes a ∈ T, b ∈ T whether a ≻ b.
Output : The set of the |T| − 1 edges of τ
n ← |T|
if n ≤ 1 then
    return {}
end
while true do
    c ← GET-PROBABLE-PSEUDO-CENTROID(T)
    (b, S) ← TEST-PSEUDO-CENTROID(T, c)
    if b = true then
        E ← ∅
        for X ∈ S do
            E ← E ∪ GET-EDGES(X)
            Let X = y_1, ..., y_m
            if c ≻ y_1 then
                E ← E ∪ {(c, max_{y∈X} y)}
            end
            else
                E ← E ∪ {(min_{y∈Anc_c} y, c)}
            end
        end
        return E
    end
end
```

Parameterized Complexity of Biclique Contraction and Balanced Biclique Contraction

R. Krithika
Indian Institute of Technology Palakkad, India

V. K. Kutty Malu
Indian Institute of Technology Palakkad, India

Roohani Sharma
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Prafullkumar Tale
Indian Institute of Science Education and Research Pune, India

Abstract

A bipartite graph is called a biclique if it is a complete bipartite graph and a biclique is called a balanced biclique if it has equal number of vertices in both parts of its bipartition. In this work, we initiate the complexity study of Biclique Contraction and Balanced Biclique Contraction. In these problems, given as input a graph G and an integer k, the objective is to determine whether one can contract at most k edges in G to obtain a biclique and a balanced biclique, respectively. We first prove that these problems are NP-complete even when the input graph is bipartite. Next, we study the parameterized complexity of these problems and show that they admit single exponential-time FPT algorithms when parameterized by the number k of edge contractions. Then, we show that Balanced Biclique Contraction admits a quadratic vertex kernel while Biclique Contraction does not admit any polynomial compression (or kernel) unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases contraction, bicliques, balanced bicliques, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.8

Funding *R. Krithika*: Supported by SERB MATRICS grant number MTR/2022/000306.

1 Introduction

Graph modification problems have been extensively studied in theoretical computer science for their expressive power. The input of a typical graph modification problem consists of a graph G and a positive integer k, and the objective is to make at most k modifications to G so that the resulting graph belongs to some specific family $\mathcal{F}$ of graphs. $\mathcal{F}$-Contraction problems refer to the variant where the only permitted modifications are edge contractions. Watanabe et al. [39] and Asano and Hirata [1] proved that if $\mathcal{F}$ is closed under edge contractions and satisfies certain specific properties, then $\mathcal{F}$-Contraction is NP-complete. Brouwer and Veldman [4] proved that $\mathcal{F}$-Contraction is NP-complete even when $\mathcal{F}$ is a singleton set consisting of a small graph like a cycle or a path on four vertices. Note that $\mathcal{F}$-Vertex Deletion and $\mathcal{F}$-Edge Deletion (the variants of the modification problems where the only allowed edits are vertex deletions and edge deletions) are trivially solvable when $\mathcal{F}$ is a fixed singleton set. This is one of the many examples that suggests that $\mathcal{F}$-Contraction problems are harder than the analogous vertex and edge deletion counterparts.

The study of graph modification problems in parameterized complexity led to the discovery of several important techniques in the field and one may argue that it played a central role in the growth of the area [2, 3, 7, 8, 9, 10, 12, 16, 17, 18, 20, 23]. The contrast in the apparent

© R. Krithika, V. K. Kutty Malu, Roohani Sharma, and Prafullkumar Tale;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 8; pp. 8:1–8:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

difficulty of contraction problems when compared to their vertex/edge deletion variants is evident even in this realm. A natural parameter for graph modification problems is the number k of allowed modifications. For a family $\mathcal{H}$ of graphs, we say that G is $\mathcal{H}$-free if for every graph $H \in \mathcal{H}$, G does not contain H as an induced subgraph. A graph family $\mathcal{F}$ *admits a forbidden set characterization* if there exists a collection $\mathcal{H}$ of graphs such that $G \in \mathcal{F}$ if and only G is $\mathcal{H}$-free. A result by Cai [5] states that if $\mathcal{F}$ is hereditary and admits a finite forbidden set characterization, then the problem of modification to $\mathcal{F}$ using any combination of vertex deletions, edge deletions and edge additions, admits a single exponential-time FPT algorithm. However, Cai and Guo [6] and Lokshtanov et al. [35] proved that $\mathcal{F}$-Contraction is W[2]-hard even when $\mathcal{F}$ admits a finite forbidden set characterization. One of the intuitive reasons for this intractability is that the classical *branching technique* that works for vertex deletion and edge deletion/addition variants does not straightaway work for contractions. Recently, Chakraborty and Sandeep [11] studied the problem of contracting to a $\mathcal{H}$-free graph where $\mathcal{H}$ is a singleton set and showed tractability and intractability results for various choices of $\mathcal{H}$.

In spite of the inherent difficulty, FPT algorithms for $\mathcal{F}$-Contraction for several graph classes $\mathcal{F}$ are known. If every graph in $\mathcal{F}$ has bounded treewidth as in the case of $\mathcal{F}$ being paths, trees, or cactus graphs, one may use Courcelle's theorem to show the existence of FPT algorithms for $\mathcal{F}$-Contraction (see [13, Chapter 7] for related definitions and arguments). For other cases of $\mathcal{F}$-Contraction, FPT algorithms have been obtained using problem-specific techniques and arguments that typically involve deep insights into the structure of $\mathcal{F}$. The first FPT algorithm for Bipartite Contraction involves an interesting combination of techniques like iterative compression, important separators, and irrelevant vertices [28] and the improved algorithm for the problem involves non-trivial applications of important separators [26]. Planar Contraction was shown to be FPT [25] using the irrelevant vertex technique combined with an application of Courcelle's theorem.

On the other hand, Clique Contraction admits a relatively simpler FPT algorithm running in $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ †† time [6]. This algorithm relies on the observation that if one can obtain a clique from a graph G by contracting k edges, then one can obtain a clique from G by deleting at most $2k$ vertices (that are endpoints of the contracted edges). That is, if (G, k) is a Yes-instance of Clique Contraction, then $V(G)$ can be partitioned into sets X and Y such that the cardinality of X is at most $2k$ and Y induces a clique. One can find such a partition (if it exists) in FPT time. Then the algorithm guesses the solution edges in $E(X) \cup E(X, Y)$ and proceeds with branching. It is surprising that this simple algorithm is optimal under the Exponential-time Hypothesis (ETH) [21]. This made us wonder if such results hold for the closely related problem of Biclique Contraction.

Biclique Contraction
Input: A graph G and an integer k.
Question: Can we contract at most k edges in G to obtain a biclique?

We call the variant of Biclique Contraction where we require the resultant graph to be a balanced biclique as Balanced Biclique Contraction. To our pleasant surprise, we are able to show that both the problems admit simple single exponential-time FPT algorithms and that neither of the problems admit subexponential-time FPT algorithms.

†† $\mathcal{O}^*(\cdot)$ notation suppresses the factors that are polynomial in the input size.

To the best of our knowledge, even the classical complexity of Biclique Contraction and Balanced Biclique Contraction is not known in the literature. As bicliques and balanced bicliques are not closed under edge contractions, one cannot use the results by Asano and Hirata [1] to show the NP-hardness of these problems. Ito et al. [30, Theorem 2] showed that for each $p, q \geq 2$, $\{K_{p,q}\}$-Contraction is NP-complete where $K_{p,q}$ denotes the biclique with p vertices in one part of the bipartition and q vertices in the other. This does not imply the NP-hardness of Biclique Contraction and Balanced Biclique Contraction. Indeed, if G is a Yes-instance of $\{K_{p,q}\}$-Contraction, then for $k = |V(G)| - (p+q)$, (G, k) is a Yes-instance of Biclique Contraction, however, the converse is not necessarily true. A similar argument holds for Balanced Biclique Contraction.

The vertex-deletion variant of the problems of deleting k vertices to get a biclique or a balanced biclique have received considerable attention in the literature. To be consistent with our terminology, we call these problems as Biclique Vertex Deletion and Balanced Biclique Vertex Deletion. Biclique Vertex Deletion is polynomial-time solvable on bipartite graphs but NP-complete in general [24, Problem GT24]. However, Balanced Biclique Vertex Deletion is NP-complete even on bipartite graphs [24, Problem GT24]. Our first result is that the analogous contraction problems are NP-complete and they remain so even on bipartite graphs.

▶ **Theorem 1.** Biclique Contraction *and* Balanced Biclique Contraction *are* NP-complete *even when the input graph is bipartite.*

We reduce Red-Blue Dominating Set to Biclique Contraction and Hypergraph 2-Coloring to Balanced Biclique Contraction to show Theorem 1. It is well-known that there are linear size-preserving reductions from 3-SAT to Red-Blue Dominating Set and from 3-SAT to Hypergraph 2-Coloring [13, 31, 38]. As the reductions used to prove Theorem 1 are also linear size-preserving, it follows that Biclique Contraction and Balanced Biclique Contraction do not admit algorithms running in $\mathcal{O}^*(2^{o(n)})$ time (and hence do not admit $\mathcal{O}^*(2^{o(k)})$ time algorithms) assuming ETH. However, as mentioned earlier, we show that both problems admit single exponential-time FPT algorithms.

▶ **Theorem 2.** Biclique Contraction *and* Balanced Biclique Contraction *can be solved in* $\mathcal{O}^*(25.904^k)$ *time.*

The only other known cases when $\mathcal{F}$-Contraction admits a single exponential-time FPT algorithm is when $\mathcal{F}$ is the collection of paths [27], trees [27], cactus [32] or grids [37]. Among these results, the algorithm for paths is relatively simple, the one for trees is obtained using non-trivial application of the color coding technique and the last two use relatively more technical problem-specific arguments. The techniques used in our algorithms include an FPT algorithm for 2-Cluster Vertex Deletion as a subroutine, rephrasing the contraction problem as a partition problem with some properties, guessing certain vertices to be in appropriate parts of this partition, preprocessing the graph based on the guess and a branching rule.

$\mathcal{F}$-Contraction for most choices of $\mathcal{F}$ mentioned here (except paths and grids) do not admit polynomial kernels. While we show Balanced Biclique Contraction to be an exception, it turns out that Biclique Contraction is not.

▶ **Theorem 3.** Balanced Biclique Contraction *admits a kernel with* $\mathcal{O}(k^2)$ *vertices. However,* Biclique Contraction *does not admit any polynomial compression or kernel unless* NP $\subseteq$ coNP*/poly.*

2 Preliminaries

For details on parameterized algorithms, we refer to standard books in the area [13, 22]. For a positive integer q, we denote the set $\{1, 2, \ldots, q\}$ by $[q]$. A partition of a set S into disjoint sets $S_1, \ldots, S_\ell$ is denoted as $\langle S_1, \ldots, S_\ell \rangle$. An *ordered partition* is one where the parts are ordered.

For an undirected graph G, $V(G)$ and $E(G)$ denote its set of vertices and edges, respectively. The *size* of a graph is the number of edges in it. A graph is *non-trivial* if it has at least one edge. We denote an edge with endpoints u and v as uv. Two vertices u and v in $V(G)$ are *adjacent* if uv is an edge in G. The *neighborhood* of a vertex v, denoted by $N_G(v)$, is the set of vertices adjacent to v. The *degree* of a vertex is the size of its neighborhood. A vertex u is a *pendant vertex* if its degree is one. We omit the subscript in the notation for neighborhood if the graph under consideration is clear. Subdividing an edge uv results in its deletion followed by the addition of a new vertex adjacent to u and v. For a subset $S \subseteq V(G)$ of vertices, $N[S] = \bigcup_{v \in S} N(v) \cup \{v\}$ and $N(S) = N[S] \setminus S$. We denote the subgraph of G induced on the set S by $G[S]$. A subset $S \subseteq V(G)$ is said to be a *connected set* if $G[S]$ is connected. For two subsets $S_1, S_2 \subseteq V(G)$, $E_G(S_1, S_2)$ denotes the set of edges with one endpoint in S_1 and the other endpoint in S_2. With a slight abuse of notation, for a set $S \subseteq V(G)$, we use $E_G(S)$ to denote $E_G(S, S)$. We say that S_1, S_2 are adjacent (or that the graphs $G[S_1]$ and $G[S_2]$ are adjacent) if $E_G(S_1, S_2) \neq \emptyset$. We omit the subscript in these notations if the graph under consideration is clear. The *disjoint union* of graphs G and H, denoted by $G + H$, is the graph with vertex set $V(H) \cup V(G)$ and edge set $E(H) \cup E(G)$ where $V(G)$ and $V(H)$ are renamed (if necessary) such that $V(G) \cap V(H) = \emptyset$. For a graph G, $\overline{G}$ denotes its complement.

A *path* is a sequence of distinct vertices in which any two consecutive vertices are adjacent. A *cycle* is a path in which the first and last vertices are adjacent. A graph is *connected* if there is a path between every pair of distinct vertices. A *component* is a maximal connected subgraph of a graph. A *spanning tree* of a connected graph is a connected acyclic subgraph which has all the vertices of the graph. A *spanning forest* of a graph is a collection of spanning trees of its components. A set of vertices is called a *clique* if any two vertices in it are adjacent. A *complete graph* is one whose vertex set is a clique and a complete graph on n vertices is denoted by K_n. A set of vertices S is said to be an *independent set* if no two vertices in S are adjacent. Throughout this paper, the input graph will be assumed to be connected.

A graph G is *bipartite* if its vertex set can be partitioned into two independent sets X, Y and $\langle X, Y \rangle$ is called a *bipartition* of G. It it well-known that a graph G is bipartite if and only if G has no odd cycles. A bipartite graph G is a *biclique* if it has a bipartition $\langle X, Y \rangle$ such that every vertex in X is adjacent to every vertex in Y. Observe that an edgeless graph is also a biclique. A biclique G with bipartition $\langle X, Y \rangle$ is a *balanced biclique* if $|X| = |Y|$. Observe that if a biclique is non-trivial, then it is connected. The following characterization of bicliques is easy to verify.

▶ **Lemma 4.** *A graph G is a biclique if and only if G is $\{K_3, K_1 + K_2\}$-free.*

Proof. ($\Rightarrow$) If G contains K_3 as an induced subgraph then G is not bipartite and hence not a biclique as well. Suppose G is a biclique with bipartition $\langle X, Y \rangle$ and has $K_1 + K_2$ as an induced subgraph. Let u, v_1, v_2 be three vertices in G such that $v_1 v_2 \in E(G)$ and $uv_1, uv_2 \notin E(G)$. Without loss of generality, let X be the part that contains v_1. Then, it follows that $v_2 \in Y$. However, $u \in X$ leads to a contradiction as $uv_2 \notin E(G)$ and $u \in Y$ also leads to a contradiction as $uv_1 \notin E(G)$.

($\Leftarrow$) Conversely, suppose G is $\{K_3, K_1 + K_2\}$-free. If G is not bipartite, then G has an odd cycle and let $C = (u_1, u_2, \ldots, u_r)$ denote the shortest odd cycle. Observe that no two non-consecutive vertices are adjacent as C is a shortest odd cycle. Then C has at least 5 vertices (as G is K_3-free) and the subgraph induced on $\{u_1, u_2, u_4\}$ is $K_1 + K_2$. Therefore, we conclude that G has no odd cycle and is hence a bipartite graph. Let $\langle X, Y\rangle$ denote a bipartition of G. If G is edgeless, then G is vacuously a biclique. Subsequently, let us assume that G is non-trivial. If G is disconnected, then it has at least two components, one of which is non-trivial. Then, the endpoints of an edge in such a non-trivial component and a vertex from another component induce a $K_1 + K_2$. Therefore, it follows that G is connected. Suppose there exist a pair of vertices $x \in X$, $y \in Y$ such that $xy \notin E(G)$. Let $z \in Y$ be a neighbour of x. Such a vertex $z \in Y$ exists as G is connected. Then, the subgraph induced on $\{x, y, z\}$ is $K_1 + K_2$ which leads to a contradiction. ◀

2.1 Edge Contractions and Graph Contractability

The *contraction* of an edge uv in G results in the addition of a new vertex that is adjacent to the vertices that are adjacent to either u or v and the subsequent deletion of the vertices u and v. The resulting graph is denoted by G/e. For a graph G and an edge $e = uv$, we formally define G/e as $V(G/e) = (V(G) \cup \{w\}) \setminus \{u, v\}$ and $E(G/e) = \{xy \mid x, y \in V(G) \setminus \{u, v\}, xy \in E(G)\} \cup \{wx \mid x \in N_G(u) \cup N_G(v)\}$ where w is a new vertex. This process does not introduce any self-loops or parallel edges. For a subset $F \subseteq E(G)$, the graph G/F denotes the graph obtained from G by contracting (in an arbitrary order) all the edges in F.

A graph G is said to be *contractible* to a graph H if there is a function $\psi : V(G) \to V(H)$ (and we say that G is contractible to H via ψ) such that the following properties hold.

- For any vertex $h \in V(H)$, $\psi^{-1}(h)$ is non-empty and connected.
- For any two vertices $h, h' \in V(H)$, $hh' \in E(H)$ if and only if $E(\psi^{-1}(h), \psi^{-1}(h')) \neq \emptyset$.

For a vertex h in H, the set $\psi^{-1}(h)$ is called a *witness set* associated with or corresponding to h. For a fixed ψ, we define the *H-witness structure* of G, denoted by $\mathcal{W}$, as the collection of all witness sets. Formally, $\mathcal{W} = \{\psi^{-1}(h) \mid h \in V(H)\}$. Note that a witness structure $\mathcal{W}$ partitions $V(G)$. If a witness set contains more than one vertex, then we call it a *big* witness set, otherwise we call it a *small* witness set or *singleton* witness set. Let $F \subseteq E(G)$ be the collection of edges of some spanning tree of $G[W]$ for each witness set $W \in \mathcal{W}$. Note that any spanning tree of the graph induced on a singleton witness set is edgeless. Now, it is sufficient to contract edges in F to obtain H from G, i.e., $G/F = H$. Hence, we say that F is a *solution* associated with the function ψ and the witness structure $\mathcal{W}$. We say G is *k-contractible* to H if there exists a subset $F \subseteq E(G)$ such that $|F| \leq k$ and $G/F = H$. Observe that in the H-witness structure of G corresponding to F, there are at most $|F|$ big witness sets and the total number of vertices in big witness sets is upper bounded by $2|F|$.

We view a biclique witness structure of G as a partition of $V(G)$ into two parts with certain properties. For a subset $X \subseteq V(G)$, let $\mathsf{sf}_G(X)$ denote the number of edges in a spanning forest of $G[X]$. We drop the subscript in the notation for $\mathsf{sf}_G(X)$ if the graph under consideration is unambigously clear.

▶ **Definition 5** (k-Constrained Valid Partition). *For a graph G, a partition $\langle L, R\rangle$ of $V(G)$ into two parts is called a k-constrained valid partition if the following properties hold.*

1. $\mathsf{sf}(L) + \mathsf{sf}(R) \leq k$.
2. *Every component of $G[L]$ is adjacent to every component of $G[R]$.*

We have the following observation on Yes-instances of Biclique Contraction.

▶ **Lemma 6.** *(G, k) is a* Yes*-instance of* Biclique Contraction *if and only if $V(G)$ has a k-constrained valid partition.*

Proof. ($\Rightarrow$) Suppose G is k-contractible to the biclique H with bipartiton $\langle X, Y\rangle$ via ψ. Let $\mathcal{W}$ denote the *H-witness structure* of G. Define subsets $\mathcal{W}_{\mathcal{L}}$ and $\mathcal{W}_{\mathcal{R}}$ of $\mathcal{W}$ as follows: $\mathcal{W}_{\mathcal{L}} = \{\psi^{-1}(h) \mid h \in V(X)\}$ and $\mathcal{W}_{\mathcal{R}} = \{\psi^{-1}(h) \mid h \in V(Y)\}$. Let L denote the collection of vertices in the witness sets in $\mathcal{W}_L$ and R denote the collection of vertices in the witness sets in $\mathcal{W}_R$. It is clear that $\langle L, R\rangle$ is a partition of $V(G)$. Let E_1 be the collection of edges that are in some spanning forest of $G[L]$ and E_2 be the collection of edges that are in some spanning forest of $G[R]$. Then, contracting E_1 in $G[L]$ results in X and contracting E_2 in $G[R]$ results in Y. Also, $E_1 \cup E_2 = E$ is a solution corresponding to $\mathcal{W}$. Then, as $|E| \leq k$, the first condition holds. As H is a biclique, every vertex of X is adjacent to every vertex of Y. Therefore, for every vertex $h \in X$ and every vertex $h' \in Y$, $E(\psi^{-1}(h), \psi^{-1}(h'))$ is not empty. In other words, for every component C_L of $G[L]$ and every component C_R of $G[R]$, there exists vertices $x \in V(C_L)$ and $y \in V(C_R)$ such that $xy \in E(G)$. Hence, the second condition holds.

($\Leftarrow$) Suppose $\langle L, R\rangle$ is a k-constrained valid partition of $V(G)$. Let E_1 be the collection of edges of some spanning forest of $G[L]$ and E_2 be the collection of edges of some spanning forest of $G[R]$. Observe that $G[L]/E_1 = X$ and $G[L]/E_2 = Y$ are independent sets. Further, every vertex in X is adjacent to every vertex in Y due to the second property. Therefore $G/(E_1 \cup E_2) = H$ is a biclique with bipartiton $\langle X, Y\rangle$. Further, as $|E_1| = k_\ell$ and $|E_2| = k_r$ with $k_\ell + k_r \leq k$ due to the first property, it follows that G is k-contractible to H. ◀

Suppose G is k-contractible to the biclique H with $\mathcal{W}$ being the H-witness structure of G and $\langle L, R\rangle$ is a k-constrained valid partition of $V(G)$ obtained from $\mathcal{W}$ as described in the proof of Lemma 6. Then, observe that the singleton witness sets in $\mathcal{W}$ correspond to the trivial components of $G[L]$ and $G[R]$ and vice-versa. We use this equivalence and the interchangeability of witness structures and constrained valid partitions throughout the paper.

In order to prove a result on Yes-instances of Balanced Biclique Contraction analogous to Lemma 6, we introduce the following definition.

▶ **Definition 7** (k-Constrained Valid Balanced Partition)**.** *For a graph G, a partition $\langle L, R\rangle$ of $V(G)$ into two parts is called a k-constrained valid balanced partition if $\langle L, R\rangle$ is a k-constrained valid partition where the number of components of $G[L]$ is equal to the number of components of $G[R]$.*

Now, we have the following property on Yes-instances of Balanced Biclique Contraction.

▶ **Lemma 8.** *(G, k) is a* Yes*-instance of* Balanced Biclique Contraction *if and only if $V(G)$ has a k-constrained valid balanced partition.*

A set $Z \subseteq V(G)$ is called a *biclique modulator* if $G - Z$ is a biclique. If G is k-contractible to a (balanced) biclique H, then there are at most $2k$ vertices that are in big witness sets of a H-witness structure. This leads to the following observation.

▶ **Observation 9.** *If G is k-contractible to a (balanced) biclique, then G has a biclique modulator of size at most $2k$.*

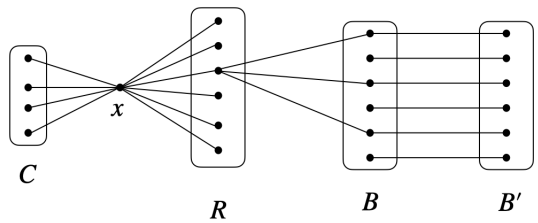


Figure 1 The graph H in the reduction from Red Blue Dominating Set to Biclique Contraction where edges between R and B are the same as in G. The vertex x is adjacent to every vertex in $R \cup C$ and each vertex in B is adjacent to exactly one vertex in B'.

3 NP-Completeness Results

In this section, we prove that Biclique Contraction and Balanced Biclique Contraction are NP-complete even when restricted to bipartite graphs.

3.1 Biclique Contraction

A set X is said to *dominate* a set Y if $Y \subseteq N(X)$. We show the NP-hardness of Biclique Contraction by giving a polynomial-time reduction from Red-Blue Dominating Set. In the Red-Blue Dominating Set problem, given a bipartite graph G with bipartition $\langle R, B\rangle$ and an integer κ, the objective is to find a set $S \subseteq R$ of size at most κ that dominates B. It is well-known and easy to verify that Red-Blue Dominating Set is equivalent to Set Cover [24, Problem SP5] and is therefore NP-hard [15].

▶ **Lemma 10.** *There is a polynomial-time reduction that takes as input an instance (G, R, B, κ) of* Red Blue Dominating Set *and returns an equivalent instance (H, k) of* Biclique Contraction *such that H is bipartite, $|V(H)| = \mathcal{O}(|V(G)|)$ and $k = |B| + \kappa$.*

Proof. Consider an instance (G, R, B, κ) of Red-Blue Dominating Set. Without loss of generality, assume $|R|, |B| > \kappa$ and that every vertex $b \in B$ is adjacent to at least two vertices in R. We construct an instance (H, k) of Biclique Contraction where $k = \kappa + |B|$ and H is obtained from G as follows.

- For every vertex $b \in B$, add a new vertex b' adjacent to b. Let B' denote the set $\{b' \mid b \in B\}$.
- Add a new vertex x adjacent to every vertex in R and add $\kappa + |B| + 1$ new vertices $v_1, \ldots, v_{\kappa+|B|+1}$ adjacent to x. Let C denote the set $\{v_1, \ldots, v_{\kappa+|B|+1}\}$.

See Figure 1 for an illustration.

It is easy to verify that the reduction takes polynomial time. Further, H is connected and bipartite with bipartition $\langle B \cup \{x\}, R \cup C \cup B'\rangle$. We show that (G, R, B, κ) is a Yes-instance of Red-Blue Dominating Set if and only if (H, k) is a Yes-instance of Biclique Contraction where $k = \kappa + |B|$.

($\Rightarrow$) Let (G, R, B, κ) is a Yes-instance of Red-Blue Dominating Set with set $S \subseteq R$ and $|S| \leq \kappa$ such that S dominates B. Then, consider the partition of $V(H)$ into sets $X = S \cup B \cup \{x\}$ and $Y = C \cup B' \cup (R \setminus S)$. Observe that $H[X]$ is connected as S dominates B and $S \subseteq N(x)$. Therefore, any spanning forest, which is also a spanning tree, of $H[X]$ has at most $\kappa + |B|$ edges. Further, Y is an independent set and since H is connected, every vertex in Y has a neighbour in X. Therefore, H/F is a biclique where F is the set of edges of some spanning tree of $H[X]$.

($\Leftarrow$) Conversely, suppose H is k-contractible to a biclique. Let $\langle X, Y\rangle$ be a k-constrained valid partition of $V(H)$ given by Lemma 6. Without loss of generality let $x \in X$. We first claim that at least one vertex from C is in Y. If $C \subseteq X$, then as $C \subseteq N(x)$ and $|C| = k+1$, any spanning forest of $H[X]$ has at least $k+1$ edges leading to a contradiction. Let $c \in C$ be a vertex in Y. Next, we claim that $H[X]$ is connected. Suppose $H[X]$ has a component $\widehat{H}$ not containing x. Then, $\widehat{H}$ has no vertex from C as $C \subseteq N(x)$ and no vertex from R as $R \subseteq N(x)$. It follows that $V(\widehat{H}) \subseteq B \cup B'$. Then, the component of $H[Y]$ containing c has no other vertex since $N(c) = \{x\}$. This implies that no vertex in the component of $H[Y]$ containing c is adjacent to a vertex in $\widehat{H}$ leading to a contradiction. Now, we argue that $\langle X, Y\rangle$ can be transformed into another k-constrained valid partition of $V(H)$ with $C \subseteq Y$. If there is a vertex $d \in X \cap C$, then as $N(d) = \{x\}$, $H[X \setminus \{d\}]$ is connected and every component of $H[Y \cup \{d\}]$ (including the trivial component in which d is in) is adjacent to $H[X \setminus \{d\}]$, $\langle X \setminus C, Y \cup C\rangle$ is the required k-constrained valid partition of $V(H)$.

Now, we transform $\langle X, Y\rangle$ into another k-constrained valid partition of $V(H)$ such that $B \subseteq X$ and $B' \subseteq Y$. Suppose there is a vertex $b \in B \cap Y$. Let $\widehat{H}$ be the component of $H[Y]$ containing b. Then, as the only neighbour of b' is b and $x \in X$, it follows that $b' \in V(\widehat{H})$. As $H[X]$ is adjacent to $\widehat{H}$, there is a vertex $r \in R$ that is adjacent to b such that $r \in V(\widehat{H})$ or $r \in X$. In the former case, we move $B \cap V(\widehat{H})$ and $R \cap V(\widehat{H})$ to X and in the latter case, we move $B \cap V(\widehat{H})$ to X. In both the cases, X remains connected and it is easy to verify that the resulting partition is a k-constrained valid partition of $V(H)$. Subsequently, we may assume $B \subseteq X$. As $H[X] \setminus B'$ is also connected and $N(B') \subseteq B$, by moving vertices from $X \cap B'$ to Y, we get another partition of $V(H)$ that is a k-constrained valid partition. Once we achieve $B \subseteq X$, we may safely move vertices of B' from X (if any) to Y. Hence, we may now assume $B \subseteq X$ and $B' \subseteq Y$.

At this point, we have $B \cup \{x\} \subseteq X$ and $B' \cap X = \emptyset$. Also, as $N(x) \cap B = \emptyset$, it follows that for each vertex $b \in B$, there is a vertex $r \in R \cap X$ in order for $H[X]$ to be connected. As any spanning tree of $H[X]$ has at most $k = \kappa + |B|$ edges, it follows that the set $R \cap X$ has at most κ vertices. Equivalently, $R \cap X$ is a set of at most κ vertices that dominates B. ◀

It is easy to verify that Biclique Contraction is in NP. This fact along with Lemma 10 establishes the first part of Theorem 1.

3.2 Balanced Biclique Contraction

We show the NP-hardness of Balanced Biclique Contraction by a reduction from Hypergraph 2-Coloring. A *hypergraph* $\mathcal{G}$ is a pair $(V, \mathcal{S})$ where V is a finite set of vertices (denoted as $V(\mathcal{G})$) and $\mathcal{S} \subseteq 2^V$ is a finite collection of subsets of V called *hyperedges*. In the Hypergraph 2-Coloring problem, the input is a hypergraph $\mathcal{G}$ and the objective is to determine if there is a 2-coloring $\phi : V(\mathcal{G}) \mapsto \{1, 2\}$ such that no hyperedge is monochromatic, i.e., a 2-coloring in which every hyperedge has a vertex with color 1 and a vertex with color 2. Hypergraph 2-Coloring is known to be NP-complete [24, Problem SP4] and is one of the natural choices picked to show the NP-hardness of contraction problems in the literature [4, 14, 19, 33].

▶ **Lemma 11.** *There is a polynomial reduction that takes as input an instance* $(\mathcal{G})$ *of* Hypergraph 2-Coloring *and returns an equivalent instance* (G, κ) *of* Balanced Biclique Contraction *such that* G *is bipartite,* $|V(G)| = \mathcal{O}(|V(\mathcal{G})|)$ *and* κ *is a function of the number of vertices and the number of hyperedges in* $\mathcal{G}$*.*

Proof. Consider an instance $(\mathcal{G} = (V, \mathcal{S}))$ of Hypergraph 2-Coloring where $V(\mathcal{G}) = \{v_1, v_2, \ldots, v_N\}$ for some $N \geq 1$ and $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$ for some $M \geq 1$. Without loss of generality, assume that $\emptyset \notin \mathcal{S}$, every hyperedge contains at least two vertices and there exists an hyperedge, say S_M, that contains all the vertices in $V(\mathcal{G})$. The reduction first constructs an intermediate non-bipartite graph H using the following procedure.

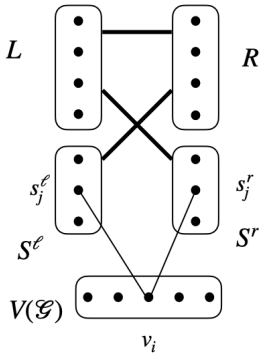


Figure 2 Construction of graph H in the reduction from Hypergraph 2-Coloring to Balanced Biclique Contraction where thick lines between two sets indicate bicliques.

- For every vertex v in $V(\mathcal{G})$, add a vertex v in H.
- For every hyperedge S_j in $\mathcal{S}$, add two vertices s_j^ℓ and s_j^r. Let $S^\ell = \{s_j^\ell \mid j \in [M]\}$ and $S^r = \{s_j^r \mid j \in [M]\}$.
- For every $i \in [N]$ and $j \in [M]$ such that $v_i \in S_j$ add edges $v_i s_j^\ell$ and $v_i s_j^r$.
- Add vertices $L = \{\ell_1, \ldots, \ell_{6M+3N-5}\}$ and $R = \{r_1, \ldots, r_{6M+3N-5}\}$ such that every vertex in L is adjacent to every vertex in R.
- Make every vertex in L adjacent to every vertex in S^r and every vertex in R adjacent to every vertex in S^ℓ.

This completes the construction of H. See Figure 2 for an illustration. We show that $(\mathcal{G})$ is a Yes-instance of Hypergraph 2-Coloring if and only if $(H, k = 2M + N - 2)$ is a Yes-instance of Balanced Biclique Contraction. The key idea behind the reduction is as follows. First, the number of vertices in L and R are so large that no edge in $E_H(L, R)$ can be contracted without exceeding the budget k. Hence, $H[L \cup R]$ acts as a representative subgraph of the final balanced biclique. Second, any contraction solution needs to partition $V(\mathcal{G})$ into two parts V_1 and V_2 such that both $H[S^\ell \cup V_1]$ and $H[S^r \cup V_2]$ are connected. This is equivalent to finding a 2-coloring of $\mathcal{G}$ such that no hyperedge is monochromatic.

($\Rightarrow$) Suppose $\phi : V(\mathcal{G}) \mapsto \{1, 2\}$ is a 2-coloring such that for no hyperedge in $\mathcal{S}$ is monochromatic. Using this coloring, we define a partition $\langle X, Y\rangle$ of $V(H)$ with $X = L \cup S^\ell \cup V_1$ and $Y = R \cup S^r \cup V_2$ where $V_1 = \{v \mid v \in V(\mathcal{G}), \phi(v) = 1\}$ and $V_2 = \{v \mid v \in V(\mathcal{G}), \phi(v) = 2\}$. As $\emptyset \notin \mathcal{S}$, $S_M = V(\mathcal{G})$, and no edge in $\mathcal{S}$ is monochromatic, $H[S^\ell \cup V_1]$ and $H[S^r \cup V_2]$ are connected. Moreover, $H[S^\ell \cup V_1]$ has a vertex adjacent to a vertex in $H[S^r \cup V_2]$. Further, as $L \cup S^\ell$, $L \cup V(\mathcal{G})$, $R \cup S^r$ and $R \cup V(\mathcal{G})$ are independent sets in H, it follows that $H[X]$ has a spanning forest of size $M + |V_1| - 1$ and $H[Y]$ has a spanning forest of size $M + |V_2| - 1$. Therefore, $\langle X, Y\rangle$ is a k-constrained valid balanced partition of $V(H)$ where $k = 2M + N - 2$. From Lemma 8, we conclude that (H, k) is a Yes-instance of Balanced Biclique Contraction.

($\Leftarrow$) Conversely, suppose H is k-contractible to a balanced biclique. Recall that an edge contraction reduces the number of vertices by exactly one. Then, H is k-contractible to the balanced biclique $K_{q,q}$ with $q \geq 6M + 3N - 4 = 3k + 2$. As in any $K_{q,q}$-witness structure of H there are at most $2k$ vertices in big witness sets and $|L| = |R| = 3k + 1$, there exist subsets $L' \subseteq L$, $R' \subseteq R$ with $|L'|, |R'| \geq k + 1$ such that every vertex of $L' \cup R'$ are in small witness sets. Now let $\langle L^*, R^*\rangle$ be a k-constrained valid balanced partition of $V(H)$ given by Lemma 8 corresponding to this witness structure. Then, $H[L^*]$ and $H[R^*]$ each have atleast $3k + 2$ components. Further, every vertex in L' is in a trivial component of $H[L^*]$ and every vertex in R' is in a trivial component of $H[R^*]$.

Next, we claim that $R \subset R^*$ and $L \subset L^*$. If there exist a vertex $r \in R \cap L^*$, then any spanning tree of $H[L' \cup \{r\}]$ has at least $k+1$ edges leading to a contradiction. A similar argument shows that $L \subset L^*$. Now we argue that every vertex of L is in a trivial component of $H[L^*]$ and every vertex of R is in a trivial component of $H[R^*]$. Suppose there are two vertices $\ell_i, \ell_j \in L$ that are in the same component X of $H[L^*]$. As ℓ_i and ℓ_j are not adjacent, there is at least one more vertex x in X that is adjacent to ℓ_i or ℓ_j. By the construction of H, $x \in R \cup S^r$. However, as $R \subset R^*$, we have $x \in S^r$. Then, any spanning tree of $H[\{x\} \cup L']$ has at least $k+1$ edges leading to a contradiction. Therefore every vertex of L is a trivial component of $H[L^*]$. A similar argument shows that every vertex of R is in a trivial component of $H[R^*]$. This shows that vertices of L form $3k+1$ components of $H[L^*]$ and vertices of R form $3k+1$ components of $H[R^*]$.

Further, as every vertex in S^r is adjacent to every vertex in L and every vertex in S^ℓ is adjacent to every vertex in R, we have $S^r \subset R^*$ and $S^l \subset L^*$. Moreover, as $L \cup V(\mathcal{G})$ and $R \cup V(\mathcal{G})$ are independent sets, we conclude that no vertex from $V(\mathcal{G})$ is in a trivial component of $H[L^*]$ or $H[R^*]$. Observe that S^ℓ is contained in one component of $H[L^*]$ if and only if S^r is contained in one component of $H[R^*]$ due to Lemma 8. Let $\langle V_1, V_2 \rangle$ be the partition of $V(\mathcal{G})$ such that $V_1 = L^* \cap V(\mathcal{G})$ and $V_2 = R^* \cap V(\mathcal{G})$. Without loss of generality, let $V_1 \neq \emptyset$. Suppose there are two components of $H[L^*]$ that contain vertices from S^ℓ. Let C' be one of these components that contain s^ℓ_M. Then, V_1 is contained in C'. Let C denote a component of $H[L^*]$ different from C' that contains vertices from S^ℓ. As $\emptyset \notin \mathcal{S}$, it follows that C has a vertex that is adjacent to a vertex in C' leading to a contradiction. Thus, $H[L^*]$ and $H[R^*]$ each have exactly $3k+2$ components implying that $H[S^\ell \cup V_1]$ and $H[S^r \cup V_2]$ are connected. Then, $\langle V_1, V_2 \rangle$ gives a 2-Coloring for $\mathcal{G}$ such that no edge is monochromatic. Hence, $(\mathcal{G})$ is a Yes-instance of Hypergraph 2-Coloring.

Now, let G be the graph obtained from H by subdividing every edge in $E_H(S^\ell, V(\mathcal{G}))$. Clearly, G is bipartite with bipartition $\langle L \cup S^\ell \cup V(\mathcal{G}), R \cup S^r \cup Z \rangle$ where Z is the set of new vertices added during the subdivision. Observe that $|Z| = \sum_{v \in V(\mathcal{G})} d(v) \geq 2M$ where $d(v)$ denotes the number of hyperedges that contain v. Then, (H, k) is a Yes-instance of Balanced Biclique Contraction if and only if $(G, k + |Z|)$ is a Yes-instance of Balanced Biclique Contraction.

Suppose H is k-contractible to the balanced biclique $K_{q,q}$. Then, from earlier arguments, we know that $q = 3k+2$. Let $\langle L^*, R^* \rangle$ be a k-constrained valid balanced partition of $V(H)$ given by Lemma 8. Then, $\langle L^* \cup Z, R^* \rangle$ is a $(k+|Z|)$-constrained valid balanced partition of $V(G)$. By Lemma 8, G is $(k+|Z|)$-contractible to a balanced biclique. Conversely, suppose G is $(k+|Z|)$-contractible to the balanced biclique $K_{p,p}$. Let $F \subseteq E(G)$ be the set of edges of G such that $G/F = K_{p,p}$. Then, as $|V(G)| = 14M + 7N - 10 + |Z|$, it follows that $2p \geq 12M + 6N - 8$ and so $p \geq 5$. As every vertex in Z has degree 2 in G, it follows that no vertex of Z is in a singleton witness set of a $K_{p,p}$-witness structure of G. In other words, each vertex of Z has at least one edge incident on it in F. For every $z \in Z$, add one such edge to F'. Then, $|F'| = |Z|$ as no edge in F has both endpoints in Z since Z is an independent set. Now, we have $G/F' = H$ and $H/(F \setminus F') = K_{p,p}$. Since $|F'| = |Z|$, it follows that $|F \setminus F'| \leq k$ and thus H is k-contractible to a balanced biclique. ◀

It is easy to verify that Balanced Biclique Contraction is in NP. This fact along with Lemma 11 establishes the second part of Theorem 1.

4 FPT Algorithms

In this section, we show that Biclique Contraction and Balanced Biclique Contraction can be solved in $\mathcal{O}^*(25.904^k)$ time. For the sake of convenience and to avoid unnecessary repetition, we first describe the algorithm for Biclique Contraction and later mention the (minor) changes required in order to solve the balanced variant.

Hüffner et al. [29] gave an $\mathcal{O}(1.4^k \cdot k^d + n^3)$ time algorithm for the d-Cluster Vertex Deletion problem in which given a graph G and integer k, the goal is to determine whether one can delete at most k vertices such that the resultant graph is the disjoint union of at most d complete graphs. As $G - Z$ is a biclique if and only if $\overline{G} - Z$ is the disjoint union of at most two complete graphs, we have the following result.

▶ **Proposition 12** ([29]). *Given a graph G and an integer k, there is an algorithm that either computes a biclique modulator Z of size at most k or correctly determines that no such modulator exists in $\mathcal{O}^*(1.4^{|Z|})$ time.*

Consider an instance (G, k) of Biclique Contraction. We first determine if G has a biclique modulator Z of size at most $2k$ in $\mathcal{O}^*(1.4^{|Z|})$ time using Proposition 12. If no such set exists, then by Observation 9, we can declare that G is not k-contractible to a biclique. Otherwise, let $\langle X, Y\rangle$ denote a bipartition of $G - Z$, and $\langle L, R\rangle$ denote a k-constrained valid partition of $V(G)$ (if one exists). We begin by guessing the "nature of the intersection" (see below for the exhaustive cases) between X, Y and L, R, and then proceed to finding a k-constrained valid partition that respects this guess. Note that if $X \cup Y$ is an empty set, then we can solve the problem by guessing the partition of Z in time $\mathcal{O}^*(2^{|Z|})$. Hence, we consider the case when $X \cup Y \neq \emptyset$. Without loss of generality, we can assume that $Y \neq \emptyset$. Consider the following cases.

1. $X \cap L = \emptyset$ and $X \cap R = \emptyset$. The two sub-cases are as follows.
   a. Either $Y \cap L \neq \emptyset$ or $Y \cap R \neq \emptyset$ but not both.
   b. $Y \cap L \neq \emptyset$ and $Y \cap R \neq \emptyset$.
2. Either $X \cap L \neq \emptyset$ or $X \cap R \neq \emptyset$ but not both. The two sub-cases are as follows.
   a. Either $Y \cap L \neq \emptyset$ or $Y \cap R \neq \emptyset$ but not both.
   b. $Y \cap L \neq \emptyset$ and $Y \cap R \neq \emptyset$.
3. $X \cap L \neq \emptyset$ and $X \cap R \neq \emptyset$. The two sub-cases are as follows.
   a. Either $Y \cap L \neq \emptyset$ or $Y \cap R \neq \emptyset$ but not both.
   b. $Y \cap L \neq \emptyset$ and $Y \cap R \neq \emptyset$.

Solving Case (1a): Without loss of generality let $Y \cap L = \emptyset$. Then, for each ordered partition $\langle Z_L, Z_R\rangle$ of Z where $L \cap Z = Z_L$ and $R \cap Z = Z_R$, we determine if $\langle Z_L, Z_R \cup Y\rangle$ is a k-constrained valid partition. We declare that G is k-contractible to a biclique if and only if some choice of $\langle Z_L, Z_R\rangle$ leads to a k-constrained valid partition. The total running time of the algorithm in this case is $\mathcal{O}^*(2^{|Z|})$.

Solving Case (1b): Consider an ordered partition $\langle Z_L, Z_R\rangle$ of Z where $L \cap Z = Z_L$ and $R \cap Z = Z_R$. Observe that as $X = \emptyset$, G is connected and Y is an independent set, every vertex in Y is adjacent to some vertex in Z_L or Z_R. Next, we apply the following branching rule repeatedly to vertices in Y as long as possible.

▶ **Branching Rule 1.** *If there is a vertex $v \in Y$ such that $N(v) \cap Z_L \neq \emptyset$, $N(v) \cap Z_R \neq \emptyset$ and $|N(v) \cap (Z_L \cup Z_R)| > 2$, then branch into the following.*

- *Contract all edges in $E(v, Z_L)$ and decrease k by the number of contractions.*
- *Contract all edges in $E(v, Z_R)$ and decrease k by the number of contractions.*

The exhaustiveness (and hence the correctness) of Branching Rule 1 is easy to verify. Also, observe that the branching factor leading to worst-case running time on applying this rule is $(1, 2)$. After an exhaustive application of Branching Rule 1, any vertex in Y that is adjacent to both Z_L and Z_R has degree 2. Then, we apply the following reduction rule.

▶ **Preprocessing Rule 1.** *If there is a vertex $v \in Y$ of degree 2 such that $N(v) \cap Z_L \neq \emptyset$ and $N(v) \cap Z_R \neq \emptyset$, then contract edges in $E(v, Z_L)$ and decrease k by the number of contractions.*

To argue correctness of this rule, it suffices to show that if $v \notin L$, then $\langle L \cup \{v\}, R \setminus \{v\}\rangle$ is also a k-constrained valid partition. Let $a \in Z_L$ and $b \in Z_R$ be the neighbours of v. Let C_a and C_b denote the components of $G[L]$ and $G[R]$ containing a and b, respectively. Note that v is a pendant vertex in C_b. The only pair of components of $G[L]$ and $G[R]$ that are adjacent possibly because of v being in R are C_a and C_b. Moving v from C_b to C_a does not affect this adjacency or the connectedness of these components since v has exactly 1 neighbour in both $V(C_b)$ and $V(C_a)$. The sizes of the spanning forests of $G[L]$ and $G[R]$ before and after the movement of v are equal. Hence, $\langle L \cup \{v\}, R \setminus \{v\}\rangle$ is also a k-constrained valid partition.

When neither Branching Rule 1 nor Preprocessing Rule 1 are applicable, we have the partition $\langle Y^L, Y^R\rangle$ of Y where $Y^L = \{y \in Y \mid N(y) \subseteq Z_L\}$ and $Y^R = \{y \in Y \mid N(y) \subseteq Z_R\}$. Let us first consider the case when no vertex in $Y_L \cup Y_R$ is a trivial component of $G[L]$ or $G[R]$. Then, observe that every vertex in Y_L has to be in the same part as Z_L since $N(Y_L) \subseteq Z_L$. Similarly, every vertex in Y_R has to be in the same part as Z_R since $N(Y_R) \subseteq Z_R$. In this case, we simply check if $(Z_L \cup Y_L, Z_R \cup Y_R)$ is a k-constrained valid partition. The total running time of the algorithm is $\mathcal{O}^*(2^{|Z|}\})$. Subsequently, we assume that there is a vertex in Y^L or Y^R that is a trivial component of $G[L]$ or $G[R]$. Without loss of generality let there be a vertex in Y^L that is a trivial component of $G[L]$ or $G[R]$. A vertex y of Y^L that is a trivial component of $G[L]$ or $G[R]$ has to be in the same part as Z_R since y has neighbours in Z_L. Further, no vertex y' in Y^R can be a trivial component of $G[L]$ or $G[R]$ and such a vertex y' has to be in the same part as Z_R. This is because if y' is in L, then the component of $G[L]$ containing y' is not adjacent to the component of $G[R]$ that contains y. On the other hand if y' is in R, then as y' is adjacent to Z_R, it cannot be a trivial component of $G[R]$. Now, it follows that every vertex in Y^R is in the same part as Z_R as no vertex in Y^R has neighbours in Z_L. Therefore, we contract all edges in $E(Y^R, Z^R)$ and decrease k by the number of contractions. Next, we guess a subset $Z' \subseteq Z_L$ that will be in a component of $G[L]$ that has no other vertices from Z_L. There are $\mathcal{O}(2^{|Z|})$ such subsets. Note that if Z' is empty, then no two vertices of Z_L are in the same component of $G[L]$ or $G[R]$. Consider the partition of Y^L into three parts Y^1, Y^2, Y^3 where $Y^1 = \{y \in Y \mid N(y) \subseteq Z'\}$, $Y^2 = \{y \in Y \mid N(y) \cap Z' = \emptyset\}$ and $Y^3 = Y \setminus (Y^1 \cup Y^2)$. Then, $Y^1 \subseteq L$ as no vertex in Y^1 is a trivial component and $N(Y^1) \subseteq Z'$. Also, $Y^2 \subseteq L$ as no vertex in Y^2 has a neighbour in Z_R or a neighbour in Z'. Similarly, $Y^3 \subseteq R$ as every vertex in Y^3 has a neighbour in Z' and a neighbour outside Z'. Therefore, we simply check if $(Z_L \cup Y^1 \cup Y^2, Z_R \cup Y^3)$ is a k-constrained valid partition. The total running time of the algorithm in this case is $\mathcal{O}^*(1.619^k 2^{2|Z|})$.

Solving Case (2a): Without loss of generality let $X \cap R = \emptyset$. Suppose $Y \cap L = \emptyset$. Then, for each ordered partition $\langle Z_L, Z_R\rangle$ of Z where $L \cap Z = Z_L$ and $R \cap Z = Z_R$, we simply determine if $\langle Z_L \cup X, Z_R \cup Y\rangle$ is a k-constrained valid partition. The total running time in this case is $\mathcal{O}^*(2^{|Z|})$. Now, suppose $Y \cap R = \emptyset$. Then, for each ordered partition $\langle Z_L, Z_R\rangle$ of Z where $L \cap Z = Z_L$ and $R \cap Z = Z_R$, we simply determine if $\langle Z_L \cup X \cup Y, Z_R\rangle$ is a k-constrained valid partition. The total running time in this case is $\mathcal{O}^*(2^{|Z|})$.

Solving Case (2b): Without loss of generality let $X \cap R = \emptyset$. Consider an ordered partition $\langle Z_L, Z_R\rangle$ of Z where $L \cap Z = Z_L$ and $R \cap Z = Z_R$. Guess a vertex $y \in Y \cap R$. There are at most n choices for y. Contract $E(y, X) \cup E(y, Z_L)$ and decrease k by the number of contractions. Observe that now we have $|X| = 1$. Then, this case is similar to Case 1(b) after moving the vertex in X to Z. The total running time in this case is $\mathcal{O}^*(1.619^k \cdot 2^{2|Z|})$.

Solving Case (3a): This is similar to Case (2b).

Solving Case (3b): In this case, all edges in $E(X, Y)$ except one gets contracted. Hence, if $|X \cup Y| > k+2$, we declare that (G, k) is a No-instance. Otherwise, $|V(G)| \leq |Z| + k + 2$. Then, we go over each ordered partition of $V(G)$ into two parts $\langle Q_L, Q_R \rangle$ and determine if $\langle Q_L, Q_R \rangle$ is a k-constrained valid partition. The running time in this case is $\mathcal{O}^*(2^{|Z|+k})$.

Now, we have the following result.

▶ **Lemma 13.** *Given a graph G and a biclique modulator Z, there is an algorithm that determines if G is k-contractible to a biclique or not in $\mathcal{O}^*(4^{|Z|}1.619^k)$ time.*

As Z can be obtained in $\mathcal{O}^*(1.4^{|Z|})$ time where $|Z| \leq 2k$, the claimed running time of $\mathcal{O}^*(25.904^k)$ to solve Biclique Contraction follows establishing part 1 of Theorem 2.

The only changes required in the algorithm for Balanced Biclique Contraction are in the places where a partition of the graph into two parts is verified if it is a k-constrained valid partition or not. This check has to modified such that the verification is made to determine if the partition is a k-constrained valid balanced partition or not. The remaining parts of the algorithm remain as such. This establishes part 2 of Theorem 2.

5 Kernelization Complexity

We begin by observing that the reduction from Red-Blue Dominating Set to Biclique Contraction described in Section 3.1 is a polynomial parameter transformation as it maps an instance (G, R, B, κ) to $(H, k = |B| + \kappa)$ where $|B| + \kappa$ is the parameter of the input instance and k is the parameter of the output instance. Using the incompressibility and infeasibility of polynomial-size kernels (with respect to $|B| + \kappa$ as the parameter) result known for Red-Blue Dominating Set [13, 15], we obtain part 2 of Theorem 3. In fact, we can also conclude that Biclique Contraction does not even admit a polynomial lossy kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ [36].

We proceed to describing a quadratic vertex kernel for Balanced Biclique Contraction using a sequence of reduction rules. The reduction rules are ordered (in the sequence stated) and a rule is applied on the instance only when none of the earlier reduction rules are applicable. Consider an instance (G, k). Let $\mathcal{C}$ be a maximal collection of vertex-disjoint $K_1 + K_2$ and K_3 in G. Let $Z = \bigcup_{C \in \mathcal{C}} V(C)$.

▶ **Reduction Rule 1.** *If $k \leq 0$ and G is not a balanced biclique or if $|Z| > 6k$, then return a trivial No-instance.*

The correctness of the first part of the rule is easy to verify. Consider the second part. Suppose (G, k) is a Yes-instance. Then, from Observation 9, there is a set $\widehat{Z}$ of at most $2k$ vertices such that $G - \widehat{Z}$ is a biclique. Further, from Lemma 4, $G - \widehat{Z}$ is $\{K_1 + K_2, K_3\}$-free. As $\widehat{Z} \cap V(S) \neq \emptyset$ for each $S \in \mathcal{C}$, it follows that $|\mathcal{C}| \leq 2k$. Then, this implies that $|Z| \leq 6k$.

Let $\langle X, Y \rangle$ be a bipartition of $G - Z$ where $|X| \leq |Y|$. Subsequently, we assume that $|Y| \geq k + 3$. Otherwise, we have a linear vertex kernel.

▶ **Reduction Rule 2.** *If $|Y| > |X| + |Z| + k$, then return a trivial No-instance.*

▶ **Lemma 14.** *Reduction Rule 2 is safe.*

Proof. We will show that if (G, k) is a Yes-instance, then $|Y| \leq |X| + |Z| + k$. Suppose G is k-contractible to the balanced biclique G/F and $\mathcal{W}$ is the G/F-witness structure of G. Let $\mathcal{W}_L$ and $\mathcal{W}_R$ be the collection of witness sets corresponding to the bipartition $\langle L, R \rangle$ of G/F.

Note that $|\mathcal{W}_L| = |\mathcal{W}_R|$. As $|F| \leq k$ and Y is an independent set in G, there are at most k vertices in Y that are incident on some edges in F. Let Y' be the collection of all such vertices. As $|Y| \geq k+1$, $Y \setminus Y'$ is a non-empty set. Note that every vertex in $Y \setminus Y'$ is in a singleton witness set in $\mathcal{W}$. As there is no edge between any of these singleton witness sets, all of these vertices are either in $\mathcal{W}_L$ or in $\mathcal{W}_R$. Without loss of generality, let every vertex in $Y \setminus Y'$ be in a witness set in $\mathcal{W}_R$. Then, $|\mathcal{W}_R| \geq |Y| - |Y'| \geq |Y| - k$. Further, there is no singleton witness set in $\mathcal{W}_L$ that contains a vertex in Y since such a witness set cannot be adjacent to a singleton witness set in $\mathcal{W}_R$ that has a vertex in $Y \setminus Y'$. This implies that every witness set in $\mathcal{W}_L$ contains at least one vertex from $X \cup Z$. Hence, $|X| + |Z| \geq |\mathcal{W}_L|$. However, $|\mathcal{W}_L| = |\mathcal{W}_R| \geq |Y| - k$ which implies $|X| + |Z| \geq |Y| - k$. ◀

Now, we have the following important property on G: if (G, k) is a Yes-instance then, in any k-constrained valid balanced partition $\langle L, R\rangle$ of $V(G)$ either $X \subseteq L$, $Y \subseteq R$ or $X \subseteq R$, $Y \subseteq L$. Suppose there are vertices $x_1, x_2 \in X$ such that $x_1 \in L$ and $x_2 \in R$. If $Y \subseteq L$, then since $|Y| \geq k+1$ the size of a spanning forest of $G[L]$ exceeds k. Similarly, if $Y \subseteq R$, then the size of a spanning forest of $G[R]$ exceeds k. Therefore, there are vertices $y_1, y_2 \in Y$ such that $y_1 \in L$ and $y_2 \in R$. Let $|X \cap L| = \alpha_X$, $|X \cap R| = \beta_X$, $|Y \cap L| = \alpha_Y$ and $|Y \cap R| = \beta_Y$. Then, any spanning forest of $G[L]$ has $k_\ell \geq \alpha_X + \alpha_Y - 1$ edges and any spanning forest of $G[R]$ has $k_r \geq \beta_X + \beta_Y - 1$ edges with $k_\ell + k_r > k$.

Let $Z_X = \{z \in Z : |N(z) \cap Y| \geq k+1\}$, $Z_Y = \{z \in Z : |N(z) \cap X| \geq k+1\}$ and $Z' = Z \setminus (Z_X \cup Z_Y)$. Observe that if $Z_X \cap Z_Y \neq \emptyset$, then (G, k) is a No-instance. Suppose there is a vertex $z \in Z_X \cap Z_Y$ and (G, k) is a Yes-instance with $\langle L, R\rangle$ being a k-constrained valid balanced partition of $V(G)$. Without loss of generality let $X \subseteq L$, $Y \subseteq R$. However, as z has at least $k+1$ neighbours each in X and Y, z cannot be in the part containing X or in the part containing Y leading to a contradiction. Hence, we may assume that the sets Z', Z_X and Z_Y partition Z. The next reduction rule is a simplification rule based on this partition.

▶ **Reduction Rule 3.** *If there is an edge in $E(X, Z_X) \cup E(Y, Z_Y) \cup E(Z_X) \cup E(Z_Y)$, then contract it and decrease k by 1.*

During the contraction of an edge incident on a vertex v in $X \cup Y$ in the application of Reduction Rule 3, the new vertex added in the process is renamed as v and retained in $X \cup Y$. Observe that the resulting graph $G[X \cup Y]$ is also a biclique.

▶ **Lemma 15.** *Reduction Rule 3 is safe.*

Proof. Suppose (G, k) is a Yes-instance and $\langle L, R\rangle$ is a k-constrained valid balanced partition of $V(G)$. Without loss of generality, $X \subseteq L$, $Y \subseteq R$. Since any $z \in Z_X$ has at least $k+1$ neighbours in Y, if $z \in R$, then any spanning forest of $G[R]$ has at least $k+1$ edges leading to a contradiction. Similarly, any $z \in Z_Y$ has at least $k+1$ neighbours in X and if $z \in L$, then any spanning forest of $G[L]$ has at least $k+1$ edges leading to a contradiction. Therefore, $Z_X \subseteq L$ and $Z_Y \subseteq R$. This justifies contracting an edge in $E(X, Z_X) \cup E(Y, Z_Y) \cup E(Z_X) \cup E(Z_Y)$. ◀

Now, we have the following property on G: if (G, k) is a Yes-instance then, in any k-constrained valid balanced partition $\langle L, R\rangle$ of $V(G)$, we have $X \cup Z_X \subseteq L$ and $Y \cup Z_Y \subseteq R$. The final reduction rule marks certain essential vertices and deletes the non-essential ones.

▶ **Reduction Rule 4.** *Mark all vertices in Z, $N(Z') \cap X$ and $N(Z') \cap Y$. Further, for each vertex $z \in Z$, mark one of its non-neighbour (if it exists) each in $X \setminus N(Z')$ and $Y \setminus N(Z')$. After performing this marking scheme, if there are at least two unmarked vertices in X and at least two unmarked vertices in Y, then delete two unmarked vertices $u \in X$ and $v \in Y$.*

The correctness of this rule is justified by the following lemma.

▶ **Lemma 16.** *Reduction Rule 4 is safe.*

Proof. Suppose G is k-contractible to a balanced biclique. Consider a k-constrained valid balanced partition $\langle L, R\rangle$ of $V(G)$. Then, we know that $X \cup Z_X \subseteq L$ and $Y \cup Z_Y \subseteq R$. As $N(u) \subseteq Y \cup Z_Y$ and $N(v) \subseteq X \cup Z_X$, it follows that u and v are in trivial components of $G[L]$ and $G[R]$, respectively. Then, $\langle L \setminus \{u\}, R \setminus \{v\}\rangle$ is a k-constrained valid balanced partition of $V(G) \setminus \{u, v\}$.

Conversely, consider a k-constrained valid balanced partition $\langle L', R'\rangle$ of $V(G) \setminus \{u, v\}$. Let G' denote $G - \{u, v\}$. Observe that $X \setminus \{u\} \subseteq L'$ and $Y \setminus \{v\} \subseteq R'$. We show that $\langle L' \cup \{u\}, R' \cup \{v\}\rangle$ is a k-constrained valid balanced partition of $V(G)$. Suppose u is in a trivial component of $G[L' \cup \{u\}]$ and is not adjacent to some component H of $G[R' \cup \{v\}]$. Then, $V(H) \subseteq Z' \cup Z_Y$ and let $z \in V(H)$. Let u^* be a non-neighbour of z in $X \setminus N(Z')$ that was marked. Since $N(u^*) \subseteq Y \cup Z_Y$, u^* is in a trivial component of $G'[L']$. In order for the component of $G'[R']$ containing z to be adjacent to the component of $G'[L']$ containing u^*, z must be in a component that contains some vertex in Y. Then, since $Y \subseteq N(u)$, this leads to a contradiction. A symmetric argument holds when v is in a trivial component of $G[R' \cup \{v\}]$ and is not adjacent to some component of $G[L' \cup \{u\}]$. Therefore, $\langle L' \cup \{u\}, R' \cup \{v\}\rangle$ is a k-constrained valid balanced partition of $V(G)$. ◀

Observe that after the application of Reduction Rule 4, there are $\mathcal{O}(k)$ unmarked vertices. Further, the marking rule procedure marks $\mathcal{O}(k^2)$ vertices. This establishes Theorem 3.

6 Concluding Remarks

In this work, we initiated the study of Biclique Contraction and Balanced Biclique Contraction. We showed NP-completeness, fixed-parameter tractability and kernelization results for the problems. A natural future direction is to study (Balanced) Biclique Contraction with respect to the size ℓ of the target (balanced) biclique as the parameter. Note that the parameterized complexity of Balanced Biclique Vertex Deletion with respect to the number ℓ of vertices in the resultant balanced biclique was a long-standing open problem until it was shown to be W[1]-hard in [34]. However, the simple reduction that takes an instance (H, k) of Independent Set and constructs an instance (G, ℓ) of Biclique Contraction where G is obtained by adding a universal vertex to H leads to the following result.

▶ **Theorem 17.** Biclique Contraction *parameterized by the lower bound $\ell = n - k$ on the number of vertices in the resultant biclique is* W[1]*-hard.*

Observe that if G is contractible to a biclique on $2 \cdot \ell$ vertices, then G is also contractible to a biclique on ℓ vertices. Hence, we can guess $\ell_0 \in \{\ell, \ell+1, \ldots, 2\cdot\ell\}$ where ℓ_0 is the smallest integer greater than or equal to ℓ such that G is contractable to a biclique on ℓ_0 vertices. However, we could not obtain a simple algorithm running in $n^{f(\ell_0)}$ time for determining if G can be contracted to a biclique on ℓ_0 vertices. In contrast, if ℓ_0 is the smallest integer greater than or equal to ℓ such that G can be contracted into a balanced biclique on ℓ_0 vertices, then there is no such easy upper bound on ℓ_0. Hence, we conjecture that Balanced Biclique Contraction is para-NP-hard when parameterized by the lower bound $\ell = n - k$ on the number of vertices in the resultant balanced biclique.

References

1. Takao Asano and Tomio Hirata. Edge-contraction problems. *J. Comput. Syst. Sci.*, 26(2):197–208, 1983. doi:10.1016/0022-0000(83)90012-0.
2. Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michal Pilipczuk. A subexponential parameterized algorithm for proper interval completion. *SIAM J. Discret. Math.*, 29(4):1961–1987, 2015. doi:10.1137/140988565.
3. Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michal Pilipczuk. Subexponential parameterized algorithm for interval completion. *ACM Trans. Algorithms*, 14(3):35:1–35:62, 2018. doi:10.1145/3186896.
4. Andries E. Brouwer and Henk Jan Veldman. Contractibility and np-completeness. *J. Graph Theory*, 11(1):71–79, 1987. doi:10.1002/jgt.3190110111.
5. 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.
6. Leizhen Cai and Chengwei Guo. Contracting few edges to remove forbidden induced subgraphs. In *Proceedings of the 8th International Symposium on Parameterized and Exact Computation, pp. 97-109*, 2013.
7. Yixin Cao. Linear recognition of almost interval graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1096–1115. SIAM, 2016. doi:10.1137/1.9781611974331.ch77.
8. Yixin Cao. Unit interval editing is fixed-parameter tractable. *Inf. Comput.*, 253:109–126, 2017. doi:10.1016/j.ic.2017.01.008.
9. Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(3):21:1–21:35, 2015. doi:10.1145/2629595.
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. Dipayan Chakraborty and R. B. Sandeep. Contracting edges to destroy a pattern: A complexity study. In Henning Fernau and Klaus Jansen, editors, *Fundamentals of Computation Theory – 24th International Symposium, FCT 2023, Trier, Germany, September 18-21, 2023, Proceedings*, volume 14292 of *Lecture Notes in Computer Science*, pages 118–131. Springer, 2023. doi:10.1007/978-3-031-43587-4_9.
12. 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.
13. 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.
14. Konrad K. Dabrowski and Daniël Paulusma. Contracting bipartite graphs to paths and cycles. *Inf. Process. Lett.*, 127:37–42, 2017. doi:10.1016/j.ipl.2017.06.013.
15. Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014. doi:10.1145/2650261.
16. Pål Grønås Drange, Markus Fanebust Dregi, Daniel Lokshtanov, and Blair D. Sullivan. On the threshold of intractability. *J. Comput. Syst. Sci.*, 124:1–25, 2022. doi:10.1016/j.jcss.2021.09.003.
17. Pål Grønås Drange, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. Exploring subexponential parameterized complexity of completion problems. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 288–299. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.288.
18. Pål Grønås Drange and Michal Pilipczuk. A polynomial kernel for trivially perfect editing. *Algorithmica*, 80(12):3481–3524, 2018. doi:10.1007/s00453-017-0401-6.

19 Jirí Fiala, Marcin Kaminski, and Daniël Paulusma. A note on contracting claw-free graphs. *Discret. Math. Theor. Comput. Sci.*, 15(2):223–232, 2013. doi:10.46298/dmtcs.605.

20 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. Syst. Sci.*, 80(7):1430–1447, 2014. doi:10.1016/j.jcss.2014.04.015.

21 Fedor V. Fomin, Daniel Lokshtanov, Ivan Mihajlin, Saket Saurabh, and Meirav Zehavi. Computation of hadwiger number and related contraction problems: Tight lower bounds. *ACM Trans. Comput. Theory*, 13(2):10:1–10:25, 2021. doi:10.1145/3448639.

22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing.* Cambridge University Press, 2019.

23 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.

24 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

25 Petr A. Golovach, Pim van 't Hof, and Daniël Paulusma. Obtaining planarity by contracting few edges. *Theor. Comput. Sci.*, 476:38–46, 2013. doi:10.1016/j.tcs.2012.12.041.

26 Sylvain Guillemot and Dániel Marx. A faster FPT algorithm for bipartite contraction. *Inf. Process. Lett.*, 113(22-24):906–912, 2013. doi:10.1016/j.ipl.2013.09.004.

27 Pinar Heggernes, Pim van 't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. *Algorithmica*, 68(1):109–132, 2014. doi:10.1007/s00453-012-9670-2.

28 Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. *SIAM J. Discret. Math.*, 27(4):2143–2156, 2013. doi:10.1137/130907392.

29 Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010.

30 Takehiro Ito, Marcin Kaminski, Daniël Paulusma, and Dimitrios M. Thilikos. Parameterizing cut sets in a graph by the number of their components. *Theor. Comput. Sci.*, 412(45):6340–6350, 2011. doi:10.1016/j.tcs.2011.07.005.

31 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

32 R. Krithika, Pranabendu Misra, and Prafullkumar Tale. An FPT algorithm for contraction to cactus. In *Proceedings of the 24th International Conference on Computing and Combinatorics, pp. 341-352*, 2018.

33 R. Krithika, Roohani Sharma, and Prafullkumar Tale. The complexity of contracting bipartite graphs into small cycles. *CoRR*, abs/2206.07358, 2022. doi:10.48550/arXiv.2206.07358.

34 Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018. doi:10.1145/3212622.

35 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. On the hardness of eliminating small induced subgraphs by contracting edges. In *Proceedings of the 8th International Symposium on Parameterized and Exact Computation, pp. 243-254*, 2013.

36 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 224–237. ACM, 2017. doi:10.1145/3055399.3055456.

37 Saket Saurabh, Uéverton dos Santos Souza, and Prafullkumar Tale. On the parameterized complexity of grid contraction. In *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT*, volume 162 of *LIPIcs*, pages 34:1–34:17, 2020. doi:10.4230/LIPIcs.SWAT.2020.34.

38 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. doi:10.1145/800133.804350.

39 Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discret. Appl. Math.*, 3(2):151–153, 1981.

Towards Identity Testing for Sums of Products of Read-Once and Multilinear Bounded-Read Formulae

Pranav Bisht
Computer Science Department, Boston College, Chestnut Hill, MA, USA
Department of Computer Science and Engineering, IIT(ISM) Dhanbad, India

Nikhil Gupta
Computer Science Department, Boston College, Chestnut Hill, MA, USA

Ilya Volkovich
Computer Science Department, Boston College, Chestnut Hill, MA, USA

Abstract

An arithmetic formula is an arithmetic circuit where each gate has fan-out one. An *arithmetic read-once formula* (ROF in short) is an arithmetic formula where each input variable labels at most one leaf. In this paper we present several efficient blackbox *polynomial identity testing* (PIT) algorithms for some classes of polynomials related to read-once formulas. Namely, for polynomial of the form:

- $f = \Phi_1 \cdot \ldots \cdot \Phi_m + \Psi_1 \cdot \ldots \cdot \Psi_r$, where Φ_i, Ψ_j are ROFs for every $i \in [m], j \in [r]$.
- $f = \Phi_1^{e_1} + \Phi_2^{e_2} + \Phi_3^{e_3}$, where each Φ_i is an ROF and e_i-s are arbitrary positive integers.

Earlier, only a whitebox polynomial-time algorithm was known for the former class by Mahajan, Rao and Sreenivasaiah (Algorithmica 2016).

In the same paper, they also posed an open problem to come up with an efficient PIT algorithm for the class of polynomials of the form $f = \Phi_1^{e_1} + \Phi_2^{e_2} + \ldots + \Phi_k^{e_k}$, where each Φ_i is an ROF and k is some constant. Our second result answers this partially by giving a polynomial-time algorithm when $k = 3$. More generally, when each Φ_1, Φ_2, Φ_3 is a multilinear bounded-read formulae, we also give a quasi-polynomial-time blackbox PIT algorithm.

Our main technique relies on the *hardness of representation* approach introduced in Shpilka and Volkovich (Computational Complexity 2015). Specifically, we show hardness of representation for the resultant polynomial of two ROFs in our first result. For our second result, we lift hardness of representation for a sum of three ROFs to sum of their powers.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Identity Testing, Derandomization, Bounded-Read Formulae, Arithmetic Formulas

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.9

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2023/109/`

Acknowledgements The authors would like to thank the anonymous referees for their detailed comments and suggestions on the previous version of the paper.

1 Introduction

Polynomial Identity Testing (PIT) is a central problem in the area of algebraic complexity theory. Given a multivariate polynomial in the form of an arithmetic circuit or a formula Φ, one is asked to decide whether Φ computes the identically zero polynomial, i.e. every coefficient in the monomial expansion of the polynomial computed by Φ is zero. There are

© Pranav Bisht, Nikhil Gupta, and Ilya Volkovich;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 9; pp. 9:1–9:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

two types of PIT algorithms – whitebox and blackbox. In the former, one can look inside the circuit or formula while in the latter one can only access evaluations of the formula on field points of their choice.

PIT is one of the important problems in the class BPP (actually in co-RP) for which a polynomial-time deterministic algorithm is yet to be found. The blackbox randomized algorithm for PIT is extremely simple: given an input polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d, pick any set $S \subseteq \mathbb{F}$ of size greater than d and evaluate f on a random point sampled from S^n. Declare the polynomial to be an identity if the evaluation is zero and a non-identity otherwise. If the polynomial was actually zero then this algorithm cannot err, otherwise for a non-zero input, this random evaluation can be zero with probability at most $d/|S|$ by the *Schwartz-Zippel-Demillo-Lipton* Lemma [50, 57, 16].

Derandomizing PIT is intimately tied to proving circuit lower bounds. A deterministic sub-exponential-time PIT algorithm yields either a super-polynomial Boolean or a super-polynomial arithmetic circuit lower bound [29, 28, 1]. Conversely, a super-polynomial arithmetic circuit lower bound implies a deterministic sub-exponential time PIT algorithm [29]. We refer the reader to the excellent survey [36] for a detailed exposition on this *hardness vs randomness* trade-off in the algebraic setting. PIT also finds applications in the problems of primality testing [2] and finding perfect matchings in graphs [38].

An arithmetic formula is an arithmetic circuit whose underlying graph is a tree. While derandomizing PIT for arithmetic formulae is still open, various interesting restricted classes have found efficient deterministic PIT algorithms. One such natural restriction is to bound the number of times a variable can appear in a formula. An arithmetic read-once formula (ROF in short) is a formula where each variable appears at most once. Shpilka and Volkovich considered the more general class of sum of k ROFs, where k is some constant. They devised a quasi-polynomial-time deterministic algorithm for this class in [53], which was later improved to polynomial time in [41]. An even more generalized model is a read-k formula, where every variable can appear at most k times, for some constant k. For the class of multilinear read-k formulas, [5] give a deterministic quasi-polynomial-time PIT algorithm. Note that the class of sum of k ROFs forms a strict subclass of multilinear read-k formulas.

From a single ROF to a sum of ROFs, the next model to consider is sum-of-products of ROFs. More generally, let $\mathcal{C}$ be any natural circuit class like ROFs, then one can define the class $\sum^{[k]} \prod \mathcal{C}$ which consists of polynomials of the form $f = \sum_{i=1}^{k} \prod_{j=1}^{m} f_{ij}$, where each f_{ij} belongs to the class $\mathcal{C}$. One can also define its sub-class $\sum^{[k]} \bigwedge \mathcal{C}$ where each product gate takes the same input. Namely, the class consists of polynomials of the form $f = \sum_{i=1}^{k} f_i^{e_i}$, where each $f_i \in \mathcal{C}$. The work of [44] proved lower bounds against the class $\sum^{[k]} \prod$ ROF, when k is constant and the product gates have certain fan-in restriction. For PIT, [39] designed a polynomial-time *whitebox* algorithm for the sub-class $\sum^{[2]} \prod$ ROF. In this work, we give a polynomial-time *blackbox* PIT algorithm for this model. For a constant k, PIT for the class $\sum^{[k]} \bigwedge$ ROF was left as an open problem by [39]. Here, we give a polynomial-time (quasi-polynomial-time) blackbox PIT algorithm for $\sum^{[k]} \bigwedge \mathcal{C}$, when $k = 3$ and $\mathcal{C}$ is the class of read-once (multilinear constant-read) formulas.

1.1 Motivations and Related Works

PIT for $\sum^{[k]} \prod \mathcal{C}$. One of the important results in the PIT literature is an efficient deterministic PIT algorithm for the class $\sum^{[k]} \prod \sum$, both in blackbox and whitebox setting, where k is a constant. The first subexponential PIT algorithm was given in [18]. The algorithm was in the whitebox setting and had quasi-polynomial run-time. Later, in [34], the

result was improved by presenting a polynomial-time whitebox algorithm. This was followed by a long line of work [32, 33, 47, 48, 49] which culminated in a polynomial-time blackbox algorithm.

A $\sum^{[k]}\prod\sum$ circuit over a field $\mathbb{F}$ computes polynomials of the kind $\sum_{i=1}^{k}\prod_{j=1}^{d_i}\ell_{i,j}$, where $d_i \in \mathbb{N}$ for every $i \in [k]$ and $\ell_{i,j} \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ are linear polynomials for every $i \in [k], j \in [d_i]$. One natural way to extend this result is to replace $\ell_{i,j}$'s with more general arithmetic circuits, for which efficient deterministic PIT algorithms are known. Some interesting candidates for such circuits are sparse polynomials (or $\sum\prod$ circuits), ROFs, multilinear bounded-read formulae, etc. Clearly, each of these circuit classes subsume the class of linear polynomials over $\mathbb{F}$. Polynomial-time deterministic blackbox PIT algorithms are known for the classes of sparse polynomials [35] and ROFs [41], and the class of multilinear bounded-read formulae admits a quasi-polynomial-time blackbox PIT algorithm [5].

PIT for the class $\sum^{[k]}\prod\sum\prod$ is well-studied: Polynomial-time deterministic blackbox PIT algorithms are known for (syntactically) multilinear $\sum^{[k]}\prod\sum\prod$ circuits [46], for *constant-read* $\sum\prod\sum\prod$ circuits [3, 7], and for the class $\sum^{[3]}\prod\sum\prod^{[2]}$ [43]; and a quasi-polynomial-time PIT algorithm for $\sum^{[k]}\prod\sum\prod^{[\delta]}$ was given in [17], where δ is also a constant. A deterministic sub-exponential PIT was given for the class $\sum\prod\sum\prod$ in the breakthrough result of [37] [1]. Note that there is no top fan-in restriction in their result. However a polynomial-time PIT algorithm continues to be elusive.

In this work, we explore the other route, i.e., in the direction of $\sum^{[k]}\prod$ ROF, which consists of circuits of the kind $\sum_{i=1}^{k}\prod_{j=1}^{d_i}\Phi_{i,j}$, where every $\Phi_{i,j}$ is an ROF over $\mathbb{F}$. The class of ROFs has been studied extensively in the Boolean as well as algebraic worlds. The results in the Boolean world include learning algorithms for Boolean ROFs and some structural properties of Boolean read-once functions [6, 30, 13, 14]. In the arithmetic world, we have the following results for the class of ROFs: A deterministic polynomial-time blackbox PIT algorithm [53, 41], efficient reconstruction algorithms [27, 12, 11, 10, 52, 41], quasi-polynomial-time deterministic blackbox PIT algorithms for the *orbit*[2] of ROFs [40, 45], a randomized polynomial-time reconstruction algorithm for orbits of ROFs [24], and characterization of read-once polynomials [56]. The investigation of PIT for $\sum^{[k]}\prod$ ROF might lead to the discovery of new ideas and techniques, which may be helpful in approaching PIT for general arithmetic circuits and formulae.

A polynomial-time deterministic whitebox PIT algorithm for $\sum^{[2]}\prod$ ROF was given in [39]. In this work, we give a polynomial-time deterministic blackbox PIT algorithm for $\sum^{[2]}\prod$ ROF (see Theorem 1). Our algorithm works over any field satisfying some mild condition on its size. The blackbox nature makes the problem quite non-trivial and we introduce a new tool for handling this: **0**-*irreducibility* (for more details, see Definition 5 and Observation 7). This tool could also be crucial to obtain PIT algorithms for $\sum^{[k]}\prod$ ROF, where $k \geq 3$ is a constant and for other interesting circuit classes.

PIT for $\sum^{[k]}\bigwedge\mathcal{C}$. The circuit class $\sum^{[k]}\bigwedge\mathcal{C}$ consists of arithmetic circuits of the type $\Phi_1^{e_1} + \cdots + \Phi_k^{e_k}$, where $\Phi_1, \ldots, \Phi_k \in \mathcal{C}$ and $e_1, \ldots, e_k \in \mathbb{N}$ are arbitrary. Apart from being a natural and interesting problem in itself, developing efficient PIT algorithm for this class is also important from the viewpoint of PIT for the class $\sum^{[k]}\prod\mathcal{C}$, which subsumes $\sum^{[k]}\bigwedge\mathcal{C}$.

[1] [37] gave a much more general result, which solves PIT for any bounded-depth arithmetic circuit in sub-exponential time.

[2] Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$. Then, the orbit of f is the set of polynomials $f(A\mathbf{x})$, where A varies over all $n \times n$ invertible matrices over $\mathbb{F}$.

Another reason for studying PIT algorithms for $\sum^{[k]} \bigwedge \mathcal{C}$ is that it generalizes the PIT for $\sum^{[k]} \mathcal{C}$, which is comprised of the circuits $\Phi_1 + \cdots + \Phi_k$, where $\Phi_1, \ldots, \Phi_k \in \mathcal{C}$. In this work, we instantiate $\mathcal{C}$ with the classes of ROFs and multilinear bounded-read arithmetic formulae, and take k to be equal to 3 (see Theorems 2 and 3). For the sake of discussion, let $\mathcal{C}_k$ be the class of read-k arithmetic formulae over a field $\mathbb{F}$. Although, $\sum^{[3]} \bigwedge \text{ROF}$ is contained in $\sum^{[3]} \bigwedge \mathcal{C}_k$, the reason for mentioning them separately is that in the case of ROFs, we obtain a deterministic polynomial-time blackbox PIT, whereas the time complexity of the blackbox PIT in the case of multilinear bounded-read formulae is quasi-polynomial.

A deterministic polynomial-time PIT algorithm is known for the class $\sum^{[k]} \text{ROF}$ [53, 41], which was built over the efficient PIT algorithm for the class of (single) ROFs [41]. This is a non-trivial generalization because the class of ROFs is not closed with respect to addition of ROFs. Now, the next level of generalization is to allow arbitrary powers of ROFs in the sum of k ROFs. This brings us to the class $\sum^{[k]} \bigwedge \text{ROF}$, for which obtaining efficient PIT has been mentioned as an open problem in [39]. PIT for the "bounded-depth variant" of this class has been studied: A polynomial-time blackbox PIT algorithm is given in [3] for the class of sum of powers of constantly many bounded-depth ROFs [3]. This algorithm is based on carefully exploiting the *Jacobian* of such circuits and the polynomial running time of their PIT crucially depends on the "bounded-depth nature" of the underlying formulae. The story of the class $\sum^{[k]} \bigwedge \mathcal{C}_k$ is also similar. [3] gives a polynomial-time deterministic blackbox PIT for the "bounded-depth" variant of this class. However, it is not clear to us how to extend their techniques to obtain a polynomial-time PIT for the classes $\sum^{[k]} \bigwedge \text{ROF}$ and $\sum^{[k]} \bigwedge \mathcal{C}_k$ in the arbitrary-depth setting.

1.2 Our Results

Now we start with our main results. Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an input polynomial of degree d. Our results below hold over *any* field $\mathbb{F}$ satisfying $|\mathbb{F}| > n \cdot d$. Otherwise, we assume to have access to a sufficiently large extension field. We note that the requirement for large enough field size is intrinsically necessary for any *blackbox* PIT algorithm.

Our first result is a blackbox PIT algorithm for $\sum^{[2]} \prod \text{ROF}$. It improves a previous result of [39], which gave a polynomial-time whitebox PIT for the same model.

▶ **Theorem 1** (Blackbox PIT for $\sum^{[2]} \prod \text{ROF}$)**.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of degree at most d computed as $f = \Phi_1 \cdots \Phi_m + \Psi_1 \cdots \Psi_r$, where $\Phi_1, \ldots, \Phi_m, \Psi_1, \ldots, \Psi_r$ are ROFs. Then there exists a deterministic algorithm that given blackbox access to f decides whether f is identically zero, in time* $\text{poly}(n, d)$.

▶ Remark 1. The parameters m and r used in the above theorem can be arbitrary as long as the degree of the polynomial computed by $\Phi_1 \cdot \cdots \cdot \Phi_m + \Psi_1 \cdots \Psi_r$ is at most d.

Due to limitations of space, the proof of this theorem is given in Appendix A. It is based on the high-level proof overview given in Section 1.3.

In the following two theorems we give blackbox PIT algorithms for the classes $\sum^{[3]} \bigwedge \text{ROF}$ and $\sum^{[3]} \bigwedge \mathcal{C}_k$, where $\mathcal{C}_k$ is the class of multilinear read-k arithmetic formulae (Definition 40). Although ROFs are subsumed by multilinear bounded-read formulae, we are stating different results for them since we obtain a polynomial-time PIT for $\sum^{[3]} \bigwedge \text{ROF}$, whereas the runtime

[3] In terminology of [3], such formulae are called as sum of constantly many bounded-depth *occur-once* formulae. In fact, a more general result along with other results was given in [3] - a polynomial-time deterministic blackbox PIT algorithm for the class of bounded-depth *bounded-occur* arithmetic formulae.

for the PIT algorithm for $\sum^{[3]} \bigwedge \mathcal{C}_k$ is quasi-polynomial. An interesting common thread in these results is that the time complexity of the blackbox PIT algorithm for $\sum^{[3]} \bigwedge$ ROF (similarly, $\sum^{[3]} \bigwedge \mathcal{C}_k$) is same as the blackbox PIT algorithm for $\sum^{[3]}$ ROF (respectively, $\sum^{[3]} \mathcal{C}_k$), which is strictly weaker than $\sum^{[3]} \bigwedge$ ROF (respectively, $\sum^{[3]} \bigwedge \mathcal{C}_k$).

▶ **Theorem 2** (Blackbox PIT for $\sum^{[3]} \bigwedge$ ROF)**.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of degree at most d computed as $f = \Phi_1^{e_1} + \Phi_2^{e_2} + \Phi_3^{e_3}$ where Φ_1, Φ_2, Φ_3 are ROFs and $e_1, e_2, e_3 \in \mathbb{N}$. Then there exists a deterministic algorithm that given blackbox access to f decides whether $f \equiv 0$, in time* $\mathrm{poly}(n, d)$.

▶ **Theorem 3** (Blackbox PIT for $\sum^{[3]} \bigwedge \mathcal{C}_k$)**.** *Let $k \in \mathbb{N}$ be a constant and let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of degree at most d computed as $f = \Phi_1^{e_1} + \Phi_2^{e_2} + \Phi_3^{e_3}$, where Φ_1, Φ_2, Φ_3 are multilinear read-k arithmetic formulae and $e_1, e_2, e_3 \in \mathbb{N}$. Then there exists a deterministic algorithm that given blackbox access to f decides whether $f \equiv 0$, in time $(n \cdot d)^{O(\log n)}$.*

Due to limitations of space, the proofs of these two theorems are given in Appendix B. In fact, we prove a more general result in Appendix B (see Theorem 58), which subsumes Theorems 2 and 3. For simplicity of exposition, we give a high-level proof overview of Theorem 2 in Section 1.3.2. The proof overview of Theorem 3 is exactly on the same line as that of Theorem 2.

1.3 Proof Overview and Techniques

In this section, we give the high level overviews of the proofs of Theorems 1, 2, and 3. The underlying theme of these proofs is the *hardness of representation* approach, which was first introduced in [53], where PIT algorithms for sums of constantly many ROFs were given. In its initial avatar, hardness of representation was given for sum of constantly many read-once polynomials (ROPs)[4] satisfying some technical conditions referred to as "**0**-justified" (Definition 2). Formally, let $\mathbb{F}$ be an arbitrary field and $A_1, \ldots, A_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be **0**-justified ROPs. Set $A \triangleq A_1 + \cdots + A_k \not\equiv 0$. Then for $n \geq 3k$ the monomial $x_1 \cdots x_n$ does not divide A. Hardness of representation approach also sits at the core of the PIT algorithm for the class of multilinear bounded-read arithmetic formulae given in [5]. In this paper, we work with a more general form of this approach. See Definition 25 and Fact 27 in this regard.

There is also an alternate way to view hardness of representation, which is popularly called *low-support concentration* in literature [4, 21, 20, 19, 26, 25, 54]. The idea is to choose a "nice" point $(a_1, \ldots, a_n) \in \mathbb{F}^n$ for an input polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that the *shifted* polynomial $f(x_1 + a_1, \ldots, x_n + a_n)$ has a non-zero monomial of low *support-size* (number of variables appearing in the monomial). For a polynomial with such a low-support monomial, efficient blackbox PIT is known (Fact 22). In this work, we shift by a *justifying assignment* or an *irreducibility-preserving assignment* (see Definitions 2, 5) in order to achieve hardness of representation, which in turn proves existence of a low-support monomial (Fact 27).

1.3.1 Proof Overview of Theorem 1

In Theorem 1, we give a polynomial-time blackbox PIT for $\sum^{[2]} \prod$ ROF, which consists of polynomials of the kind $f = A_1 \cdots A_m + B_1 \cdots B_r$, where every $A_\ell, B_t \in \mathbb{F}[x_1, \ldots, x_n]$ are read-once polynomials (ROPs), i.e., the polynomials computed by ROFs (Definition 28).

[4] A *read-once polynomial* is a polynomial that is computable by a read-once formula

We are given blackbox access to such an $f = A_1 \cdots A_m + B_1 \cdots B_r$, where $\deg(f) \leq d$ and we want to determine whether $f \equiv 0$ or not. To accomplish this, we design a *hitting-set generator*, that is a polynomial map $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^w \to \mathbb{F}^n$ which preserves non-zeroness, formally $f(x_1, \ldots, x_n) \equiv 0$ if and only if $f(\mathcal{G}^1, \ldots, \mathcal{G}^n) \equiv 0$, where w is a constant and $\max\{\deg(\mathcal{G}^i) : i \in [n]\} \leq \delta$. Then, $f(\mathcal{G})$ becomes a w-variate polynomial, which has degree at most $d \cdot \delta$. Since w is a constant, it is easy to test the zeroness of $f(\mathcal{G})$ in time $(nd\delta)^{O(w)}$. The map $\mathcal{G}$ in our case is the generator $G_{n,4}$ given in Definition 20, with $w = 8$.

Now, let us see why $G_{n,4}$ is a correct generator for the class $\sum^{[2]} \prod \text{ROF}$. Recall $f = A_1 \cdots A_m + B_1 \cdots B_r$. As every A_ℓ, B_t are ROPs, we can assume without loss of generality that they are irreducible (see Fact 29). We now apply the standard trick of *simplifying* the polynomial. Formally, let $g \triangleq \gcd(A_1 \cdots A_m, B_1 \cdots B_r)$ and $f' \triangleq \frac{f}{g}$. Then, we can write $f = g \cdot f'$, where g is called the *simple* part of f. Since g is a product of non-zero ROPs (see Fact 29), it follows from a result of [41] (see Fact 38) and the *multiplicative property* of a generator that $f(G_{n,4}) \equiv 0$ if and only if $f'(G_{n,4}) \equiv 0$. So, we can assume without loss of generality that $f' = f = A_1 \cdots A_m + B_1 \cdots B_r$. Then, there are two possibilities: Either $f \in \mathbb{F}$ or for every $\ell \in [m], t \in [r]$, A_ℓ and B_t are co-prime. The first case is trivial. Now, we talk about the second case.

Fix $A = A_\ell$ and $B = B_t$ arbitrarily. Since $\gcd(A, B) = 1$, if x appears in A then the resultant of A and B with respect to x, denoted $\text{Res}_x(A, B)$, is a non-zero polynomial (see Definition 9 and Fact 10). As A, B are ROPs, they are multilinear, and can be written as $A = A'_1 x + A'_0$ and $B = B'_1 x + B'_0$, where $A'_1, A'_0, B'_1, B'_0 \in \mathbb{F}[\mathbf{x} \setminus \{x\}]$. Then, it follows from Definition 9 that $\text{Res}_x(A, B) = A'_1 B'_0 - A'_0 B'_1$. Since A and B are co-prime polynomials, $\text{Res}_x(A, B) \not\equiv 0$. If we have a generator $\mathcal{G}$ that hits this resultant, then we are done as $A(\mathcal{G})$ will be co-prime to every $B(\mathcal{G})$, which certifies that $f(\mathcal{G}) \not\equiv 0$. This approach has been utilized earlier also and is formally stated in Fact 24. In order to argue that $f(G_{n,4}) \not\equiv 0$, it suffices to prove that $G_{n,3}$ hits the resultant $\text{Res}_x(A, B)$, i.e. $(\text{Res}_x(A, B))(G_{n,3}) \not\equiv 0$.

Now, we argue that $(\text{Res}_x(A, B))(G_{n,3}) \not\equiv 0$. For this, we introduce the notion of *zero-irreducibility*. We say that a polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ is $\mathbf{0}$-*irreducible*, if for every proper subset $I \subsetneq [n]$, the restricted polynomial $g|_{\mathbf{x}_I = \mathbf{0}_I}$ is irreducible and $g(\mathbf{0}) \neq 0$ (Definition 5). Let us first see why $\mathbf{0}$-irreducible ROPs are interesting in this scenario. Let $\widetilde{A}$ and $\widetilde{B}$ be two $\mathbf{0}$-irreducible ROPs and $x \in \text{var}(\widetilde{A})$. We show that there exists a monomial in $\text{Res}_x(\widetilde{A}, \widetilde{B})$, which has at most two variables (see Corollary 46). This along with Fact 22 implies that $(\text{Res}_x(\widetilde{A}, \widetilde{B}))(G_{n,2}) \not\equiv 0$. To show that a monomial of support at most two exists in $\text{Res}_x(\widetilde{A}, \widetilde{B})$, we prove a *hardness of representation* theorem for the resultant of two $\mathbf{0}$-irreducible ROPs. In particular, we show that if $\widetilde{A}, \widetilde{B}$ are $\mathbf{0}$-irreducible ROPs then there do not exist three distinct variables x_1, x_2, x_3 such that $x_1 x_2 x_3$ divides $\text{Res}_x(\widetilde{A}, \widetilde{B})$ (see Lemma 45). This result is the heart of the proof of Theorem 1.

Now let us see how to transform the original irreducible ROPs A, B to $\mathbf{0}$-irreducible ROPs $\widetilde{A}, \widetilde{B}$. We show that there exists an assignment $\mathbf{a}$ in the image of $G_{n,1}$ such that $\widetilde{A} \triangleq A(\mathbf{x} + \mathbf{a})$ and $\widetilde{B} \triangleq B(\mathbf{x} + \mathbf{a})$ are $\mathbf{0}$-irreducible ROPs. For this, we need a tool called the *commutator* of a polynomial $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ (Definition 13), denoted $\Delta_{i,j} g$, where $i, j \in [n]$. Since A, B are irreducible multilinear polynomials, all the commutators of A and B are non-zero (Corollary 17). We show that if we have an $\mathbf{a} \in \mathbb{F}^n$ such that $A(\mathbf{a}) \neq 0, B(\mathbf{a}) \neq 0$ and for every $i \neq j \in [n], (\Delta_{i,j} A)(\mathbf{a}) \neq 0, (\Delta_{i,j} B)(\mathbf{a}) \neq 0$, then $A(\mathbf{x} + \mathbf{a})$ and $B(\mathbf{x} + \mathbf{a})$ are $\mathbf{0}$-irreducible ROPs. The nice structure of a commutator of an ROP given in Corollary 34 turns out to be extremely helpful in showing that the desired tuple $\mathbf{a}$ is in the image of $G_{n,1}$.

On putting the things together, we get that $(\text{Res}_x(A, B))(G_{n,2} + G_{n,1}) \not\equiv 0$. Since $G_{n,3} = G_{n,2} + G_{n,1}$ (see Fact 21), we have $\text{Res}_x(A, B)(G_{n,3}) \not\equiv 0$.

1.3.2 Proof Overview of Theorems 2 and 3

In Section B, we prove a result (see Theorem 58) which captures both Theorems 2 and 3. However, for the sake of keeping the discussion simple and yet deliver the main ideas, we restrict ourselves to ROFs. In particular, we give a high-level proof overview of Theorem 2.

We are given blackbox access to an f computed by a circuit in $\sum^{[3]} \bigwedge$ ROF and we want to determine whether f is zero or not. Then, there exist three ROPs $A, B, R \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $e_1, e_2, e_3 \in \mathbb{N}$ such that $f = A^{e_1} + B^{e_2} + R^{e_3}$. We prove that $f \equiv 0$ if and only if $f(G_{n,10}) \equiv 0$, where $G_{n,10}$ is given in Definition 20. The main crux of this result is the hardness of representation for f. We show that if an assignment $\mathbf{a} \in \mathbb{F}^n$ is a common *justifying assignment* (Definition 2) of A, B, R then $f(\mathbf{x} + \mathbf{a})$ is either zero or for every set $J \subseteq [n]$ of size 10, $f' \triangleq f(\mathbf{x} + \mathbf{a})$ is not divisible by the monomial $\prod_{j \in J} x_j$. This hardness of representation then implies existence of a monomial of support-size at most 9 in f' and therefore $G_{n,9}$ hits f' (see Fact 27). Formally, $f' \equiv 0$ if and only if $f'(G_{n,9}) \equiv 0$. We know from [53] that such an $\mathbf{a}$ is in the image of $G_{n,1}$ (see Fact 30). Since $G_{n,10} = G_{n,9} + G_{n,1}$ (see Fact 21), we get that $G_{n,10}$ is a generator for f. Now, it is not difficult to argue that given blackbox access to f, we can test whether f is zero or not in $\mathrm{poly}(n, d)$-time, where $d = \deg(f)$.

The hardness of representation theorem mentioned above crucially uses the fact that f is a sum of powers of three ROFs. The proof proceeds by analyzing various cases originating from the comparison of the parameters e_1, e_2, and e_3 mentioned above. Here, we assume without loss of generality that $e_1 \geq e_2 \geq e_3$. If $e_1 > e_2$ then it is easy to show the required hardness of representation result. A major chunk of the proof is devoted to analyze the case when $e_1 = e_2 = e$. In this part, the following factorization becomes pivotal.

$$A^e - B^e = \prod_{\ell \in [e]} (A - \omega^\ell B),$$

where ω is a primitive e-th root of unity. It may seem from here that our result only holds over fields that contain ω. However, it is not the case. We show that it is possible to "massage" e in such a way that a primitive e-th root of unity is always present in the underlying field (or an appropriate extension). Our proof crucially exploits the following two properties of ROFs: 1) the class of ROFs is closed under product of variable disjoint ROFs, and 2) the hardness of representation result for the sum of any three $\mathbf{0}$-justified ROFs given in [53] (see Fact 35).

2 Preliminaries

For a field $\mathbb{F}$, its algebraic closure is denoted as $\overline{\mathbb{F}}$. $\mathbb{N}$ represents the set of natural numbers. For $n \in \mathbb{N}, [n] \triangleq \{1, \ldots, n\}$. Unless otherwise specified, $\mathbf{x} \triangleq \{x_1, \ldots, x_n\}$. We denote the sets of variables by $\mathbf{x}, \mathbf{y}, \mathbf{z}$; polynomials by $f, g, h, u, v, A, B, F, R$; elements of $\mathbb{F}$ by α, β, a, b; vectors over $\mathbb{F}$ by $\mathbf{a}, \mathbf{b}$; circuit classes by upper case calligraphic letters like $\mathcal{C}$; and sets by I, J, K, L. For a polynomial $f \in \mathbb{F}[\mathbf{x}]$, we denote a monomial $x_1^{e_1} \cdots x_n^{e_n}$ in f by $\mathbf{x}^{\mathbf{e}}$ and for some $i \in [n]$, $\deg_{x_i}(f)$ denotes the degree of variable x_i in f when it is viewed as a polynomial in x_i over $\mathbb{F}[\mathbf{x} \setminus \{x_i\}]$. The *individual degree* of f is defined as $\max_{i \in [n]}\{\deg_{x_i}(f)\}$. A polynomial f is called *multilinear* if its individual degree is at most one. We define support of a monomial by $\mathrm{supp}(\mathbf{x}^{\mathbf{e}}) \triangleq \{i \in [n] \mid e_i > 0\}$ and denote support-size by $|\mathrm{supp}(\mathbf{x}^{\mathbf{e}})|$.

We call $f, g \in \mathbb{F}[\mathbf{x}]$ are *similar*, denoted $f \sim g$, if there exists a non-zero $\alpha \in \mathbb{F}$ such that $f = \alpha \cdot g$. For a polynomial $f \in \mathbb{F}[\mathbf{x}]$ and a vector $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$, the shifted polynomial is $f(\mathbf{x} + \mathbf{a}) \triangleq f(x_1 + a_1, \ldots, x_n + a_n)$. We say that $f \in \mathbb{F}[\mathbf{x}]$ *depends* on x_i

if there exist $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$ and $b \in \mathbb{F}$ such that $f(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n)$. Further, $\mathrm{var}(f) \triangleq \{i \in [n] : f \text{ depends on } x_i\}$. Let $f \in \mathbb{F}[\mathbf{x}], I \subseteq [n]$, and $\mathbf{a} \in \mathbb{F}^n$. Then, $f|_{\mathbf{x}_I = \mathbf{a}_I}$ is obtained by substituting $x_i = a_i$ in f for every $i \in I$. Clearly, $\mathrm{var}(f|_{\mathbf{x}_I=\mathbf{a}_I}) \subseteq \mathrm{var}(f) \setminus I$. Observe that this containment can be strict. For example, let $f = x_1x_2 + 1, \mathbf{a} = (0, 0)$, and $I = \{1\}$. Then, $\mathrm{var}(f|_{\mathbf{x}_I=\mathbf{a}_I}) \subsetneq \mathrm{var}(f) \setminus \{1\}$ as after setting $x_1 = 0$ in f, it no longer depends on x_2. We are interested in those assignments where such undesirable losses do not happen. Such assignments, known as *justifying assignments*, have been earlier considered in [27, 12, 53]. Consider the following definition, which has been obtained by adding Property 2 to the definition of a justifying assignment given in [53, 52]. This modification has been made to suit our purpose.

▶ **Definition 2** (Justifying assignment). *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $\mathbf{a} \in \mathbb{F}^n$. Then, $\mathbf{a}$ is called a justifying assignment of f (equivalently, f is said to be $\mathbf{a}$-justified) if*

1. *for every $I \subseteq \mathrm{var}(f), \mathrm{var}(f|_{\mathbf{x}_I=\mathbf{a}_I}) = \mathrm{var}(f) \setminus I$ and*
2. *$f(\mathbf{a}) \neq 0$.*

▶ Remark 3. By convention, the identically zero polynomial is $\mathbf{a}$-justified for every $\mathbf{a} \in \mathbb{F}^n$.

The following nice property of a justifying assignment is implied by Proposition 2.3 of [53].

▶ **Fact 4.** *An assignment $\mathbf{a} \in \mathbb{F}^n$ is a justifying assignment of a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ if and only if the condition given in Property 1 of Definition 2 holds for every $I \subseteq \mathrm{var}(f)$ of size $|\mathrm{var}(f)| - 1$ and $f(\mathbf{a}) \neq 0$.*

An $f \in \mathbb{F}[\mathbf{x}] \setminus \mathbb{F}$ is *irreducible* if it can not be written as a product of two non-constant polynomials in $\mathbb{F}[\mathbf{x}]$. Otherwise, f is reducible. By convention, every element of $\mathbb{F}$ is reducible.

▶ **Definition 5** (Irreducibility preserving assignment). *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $\mathbf{a} \in \mathbb{F}^n$. Then, $\mathbf{a}$ is called an* irreducibility preserving assignment *of f if for every proper subset $I \subsetneq \mathrm{var}(f)$, the restricted polynomial $f|_{\mathbf{x}_I=\mathbf{a}_I}$ is irreducible and $f(\mathbf{a}) \neq 0$. Equivalently, we say that f is $\mathbf{a}$-irreducible.*

For example, let $f = x_1 + x_2 + x_3$ and $\mathbf{a} = (0, 0, 1)$. Then, f is $\mathbf{a}$-irreducible over every field. Observe that if $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is $\mathbf{a}$-irreducible for any $\mathbf{a} \in \mathbb{F}^n$ then f is irreducible over $\mathbb{F}$. Claim 6 below shows that irreducibility preserving assignments capture justifying assignments of irreducible polynomials. Note that the converse of this claim is not true. For example, let $f = (x_1 + 1)(x_2 + 1) + x_3$ and $\mathbf{a} = (0, 0, 0)$. Then, f is irreducible, $\mathbf{a}$-justified but is not $\mathbf{a}$-irreducible. Thus, for irreducible polynomials, the notion of irreducible preserving assignment is strictly stronger than that of justifying assignment.

▷ Claim 6 ([8]). Let $f \in \mathbb{F}[\mathbf{x}]$ and $\mathbf{a} \in \mathbb{F}^n$. If f is $\mathbf{a}$-irreducible then f is $\mathbf{a}$-justified.

▶ **Observation 7.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $\mathbf{a} \in \mathbb{F}^n$. Then, f is $\mathbf{a}$-irreducible if and only if $f(\mathbf{x} + \mathbf{a})$ is $\mathbf{0}$-irreducible.*

2.1 Basic Mathematical Facts

▶ **Fact 8** (Gauss Lemma). *Let $\mathbb{F}$ be a field, $f \not\equiv 0 \in \mathbb{F}[\mathbf{x}, y]$, and $g \in \mathbb{F}[\mathbf{x}]$. Suppose $f|_{y=g(\mathbf{x})} \equiv 0$. Then, $y - g(\mathbf{x})$ is an irreducible factor of f.*

Resultant

The polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ is a unique factorization domain (UFD) and therefore the gcd of two polynomials is well defined. One can also define gcd w.r.t. a single variable, say x_i, by viewing the polynomials as univariates in x_i, with coefficients in $\mathbb{F}[\mathbf{x} \setminus \{x_i\}]$. Then, $\gcd_{x_i}(f, g)$ is well defined up to multiplication by a rational function in $\mathbb{F}(\mathbf{x} \setminus \{x_i\})$. In this case, we work with the normalized gcd. For example, let $f = x^3y + xy^3$ and $g = xy^2$, then $\gcd(f, g) = xy$ and $\gcd_y(f, g) = y$. The former is gcd in $\mathbb{F}[x, y]$, while the latter is normalized gcd in $\mathbb{F}(x)[y]$. See [22] for details.

Let $f, g \in \mathbb{F}[x_1, \ldots, x_n, y]$ be two non-zero polynomials of y-degree d and e, respectively. Suppose $f(y) = \sum_{i=0}^{d} a_i \cdot y^i$ and $g(y) = \sum_{j=0}^{e} b_j \cdot y^j$, where each $a_i, b_j \in \mathbb{F}[x_1, x_2, \ldots, x_n]$. The *Sylvester matrix* M is the following $(d + e) \times (d + e)$ matrix

$$M = \begin{bmatrix} a_d & a_{d-1} & \ldots & a_1 & a_0 & & & \\ & a_d & a_{d-1} & \ldots & a_1 & a_0 & & \\ & & \ldots & \ldots & \ldots & \ldots & & \\ & & & a_d & a_{d-1} & \ldots & a_1 & a_0 \\ b_e & b_{e-1} & \ldots & b_1 & b_0 & & & \\ & b_e & b_{e-1} & \ldots & b_1 & b_0 & & \\ & & \ldots & \ldots & \ldots & \ldots & & \\ & & & b_e & b_{e-1} & \ldots & b_1 & b_0 \end{bmatrix}.$$

▶ **Definition 9** (Resultant). *For $f, g \in \mathbb{F}[y, x_1, \ldots, x_n]$, the resultant $\mathrm{Res}_y(f, g) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is defined as determinant of the Sylvester matrix. That is, $\mathrm{Res}_y(f, g) = \det(M)$.*

We use the following properties of the Resultant:

▶ **Fact 10** (See [23, 22, 15]). *Let $f, g \in \mathbb{F}[y, x_1, \ldots, x_n]$. Then,*

1. *$\gcd_y(f, g) \neq 1$ if and only if $\mathrm{Res}_y(f, g) \equiv 0$. That is, f and g have a non-trivial factor that depends on the variable y (i.e., $\deg_y(\gcd(f, g)) > 0$) if and only if the y-resultant of f, g is the identically zero polynomial.*
2. *Let $\mathbf{a} \in \mathbb{F}^n$. If $\deg_y(f) = \deg_y(f|_{\mathbf{x}=\mathbf{a}})$ and $\deg_y(g) = \deg_y(g|_{\mathbf{x}=\mathbf{a}})$, then $\mathrm{Res}_y(f, g)|_{\mathbf{x}=\mathbf{a}} = \mathrm{Res}_y(f|_{\mathbf{x}=\mathbf{a}}, g|_{\mathbf{x}=\mathbf{a}})$.*

2.2 Partial Derivatives

The following definition of discrete partial derivatives is taken from [53].

▶ **Definition 11** (Discrete Partial Derivative). *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $x \in \{x_1, \ldots, x_n\}$. Then, the* discrete partial derivative *of f with respect to x is defined as: $\frac{\partial f}{\partial x} \triangleq f|_{x=1} - f|_{x=0}$.*

Further, let $I = \{i_1, \ldots, i_r\} \subseteq [n]$ be a non-empty set of size. We denote by $\partial_I f$ the iterated partial derivative *of f with respect to I.*

The following fact relates partial derivatives and justifying assignments of multilinear polynomials. A polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is *multilinear* if the individual degree of every variable in f is at most one.

▶ **Fact 12** (Lemma 2.6 [53]). *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and $\mathbf{a} \in \mathbb{F}^n$. Then, $\mathbf{a}$ is a justifying assignment of f if and only if $f(\mathbf{a}) \neq 0$ and for every $x_i \in \mathrm{var}(f)$ we have that $\frac{\partial f}{\partial x_i}(\mathbf{a}) \neq 0$.*

2.3 Commutator

This section is devoted to commutators of polynomials. This tool was defined in [51], where it was used in the context of polynomial factorization. It also played a crucial role in the deterministic reconstruction algorithm for read-once formulas (Definition 28) given in [52]. The following definition of a commutator of a polynomial is taken from [52]. This can be seen as a determinant of a special case Nisan's partial derivative matrix.

▶ **Definition 13** (Commutator). *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $i \neq j \in [n]$. Then, the commutator of f with respect to x_i and x_j, denoted $\Delta_{i,j} f$, is defined as*

$$\Delta_{i,j} f = f|_{x_i=1,x_j=1} \cdot f|_{x_i=0,x_j=0} - f|_{x_i=1,x_j=0} \cdot f|_{x_i=0,x_j=1}.$$

We note that this definition of a commutator of a polynomial is different from the definition given in [51]. However, it is not difficult to show that both these definitions have same properties for multilinear polynomials. We now note some useful properties of commutators of multilinear polynomials.

▶ **Observation 14.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and $i, j \in [n]$. Then, f can be written as $f = f_{i,j} x_i x_j + f_i x_i + f_j x_j + f_0$, where $f_{i,j}, f_i, f_j, f_0 \in \mathbb{F}[\mathbf{x} \setminus \{x_i, x_j\}]$. Then $\Delta_{i,j} = f_{i,j} \cdot f_0 - f_i \cdot f_j$.*

Using this, we can easily prove the following observation, which would be used in Section 3.

▶ **Observation 15.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and $i \neq j \in [n]$ such that $\frac{\partial^2 f}{\partial x_i \partial x_j} \equiv 0$. Then, $\Delta_{i,j} f = -\frac{\partial f}{\partial x_i} \cdot \frac{\partial f}{\partial x_j}$.*

▶ **Fact 16** (Lemma 4.6 of [51]). *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a non-constant multilinear polynomial and $i \neq j \in [n]$. There exist $g, h \in \mathbb{F}[\mathbf{x}]$ where $i \notin \mathrm{var}(h)$ and $j \notin \mathrm{var}(g)$ such that $f = g \cdot h$ if and only if $\Delta_{i,j} f \equiv 0$.*

The fact above implies the following result.

▶ **Corollary 17.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial, where $n \geq 2$. Then, f is reducible if and only if there exist $i, j \in \mathrm{var}(f)$ such that $\Delta_{i,j} f \equiv 0$.*

The following property of a commutator immediately follows from Definition 13.

▶ **Observation 18.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial, $\mathbf{a} \in \mathbb{F}^n, i \neq j \in \mathrm{var}(f)$, and $I \subseteq \mathrm{var}(f) \setminus \{i, j\}$. Then, $\Delta_{i,j}(f|_{\mathbf{x}_I=\mathbf{a}_I}) = (\Delta_{i,j} f)|_{\mathbf{x}_I=\mathbf{a}_I}$.*

The following useful claim relates commutators and irreducibility preserving assignments, which would play an important role in Section 3.1.4.

▷ Claim 19 (Commutators and irreducibility preserving assignments). Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and $\mathbf{a} \in \mathbb{F}^n$ s.t. $f(\mathbf{a}) \neq 0$. Suppose that for every $i \neq j \in \mathrm{var}(f)$: $(\Delta_{i,j} f)(\mathbf{a}) \neq 0$. Then, f is $\mathbf{a}$-irreducible.

Proof. Suppose f is not $\mathbf{a}$-irreducible. Then, either $f(\mathbf{a}) = 0$ or there exists a proper subset $I \subsetneq \mathrm{var}(f)$ such that $f|_{\mathbf{x}_I=\mathbf{a}_I}$ is reducible. In the former case, we immediately get a contradiction. Now, suppose the latter holds. Then, there exist non-constant multilinear polynomials $g, h \in \mathbb{F}[\mathbf{x}]$ such that $f|_{\mathbf{x}_I=\mathbf{a}_I} = g \cdot h$. Let $i \in \mathrm{var}(g)$ and $j \in \mathrm{var}(h)$. As $f|_{\mathbf{x}_I=\mathbf{a}_I}$ is multilinear, g and h are variable disjoint. Then, it follows from Corollary 17 that $\Delta_{i,j}(f|_{\mathbf{x}_I=\mathbf{a}_I}) \equiv 0$, which implies $(\Delta_{i,j}(f|_{\mathbf{x}_I=\mathbf{a}_I}))(\mathbf{a}) = 0$. Observation 18 implies that $(\Delta_{i,j}(f|_{\mathbf{x}_I=\mathbf{a}_I}))(\mathbf{a}) = (\Delta_{i,j} f)(\mathbf{a})$. Since $(\Delta_{i,j}(f|_{\mathbf{x}_I=\mathbf{a}_I}))(\mathbf{a}) = 0$, we get $(\Delta_{i,j} f)(\mathbf{a}) = 0$, which is a contradiction. Hence, f is $\mathbf{a}$-irreducible. ◁

2.4 The Generator $G_{n,k}$ of [53]

Due to limitations of space, see [8] for formal definitions of a hitting set and a generator. Here, we discuss the specific generator $G_{n,k}$, which was defined in [53] for the class of ROFs. It has been a crucial ingredient in PIT algorithms of various other interesting classes also [31, 20, 5, 41]. We will also be using this generator in our results. We borrow the definition and properties of this generator as presented in [5, 55].

▶ **Definition 20.** *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ be n distinct elements and for $i \in [n]$, let $L_i(x) \triangleq \prod_{j\in[n]\setminus\{i\}} \frac{x-\alpha_j}{\alpha_i-\alpha_j}$ denote the corresponding Lagrange interpolant. For every $k \in [n]$, let $G_{n,k} : \mathbb{F}^{2k} \to \mathbb{F}^n$ be defined as*

$$G_{n,k}(y_1, \ldots, y_k, z_1, \ldots, z_k) \triangleq \left(\sum_{j=1}^{k} L_1(y_j)z_j, \sum_{j=1}^{k} L_2(y_j)z_j, \ldots, \sum_{j=1}^{k} L_n(y_j)z_j \right)$$

Let $(G_{n,k})_i$ denote the ith component of $G_{n,k}$ and we call α_i as the Lagrange constant *associated with this i^{th} component. We can also define G_k to be the class of generators $\{G_{n,k}\}_{n\in\mathbb{N}}$ for all output lengths.*

For two generators $\mathcal{G}_1, \mathcal{G}_2$ with the same output length, we define their sum $\mathcal{G}_1 + \mathcal{G}_2$ as their component-wise addition, where the seed variables of both generators are implicitly relabelled so as to be disjoint. With this terminology, we can note various useful properties of the generator $G_{n,k}$ from its definition.

▶ **Fact 21** ([53, 31, 55])**.** *Let k, k' be positive integers.*

1. $G_{n,k}(\mathbf{y}, \mathbf{0}) \equiv \mathbf{0}$.
2. $G_{n,k}(y_1, \ldots, y_k, z_1, \ldots, z_k)|_{y_k=\alpha_i} = G_{n,k-1}(y_1, \ldots, y_{k-1}, z_1, \ldots, z_{k-1}) + z_k \cdot \mathbf{e}_i$, *where $\mathbf{e}$ is the 0-1 vector with a single 1 in coordinate i and α_i the ith Lagrange constant and $G_{n,0} \triangleq \mathbf{0}$.*
3. $G_{n,k}(y_1, \ldots, y_k, z_1, \ldots, z_k) + G_{n,k'}(y_{k+1}, \ldots, y_{k+k'}, z_{k+1}, \ldots, z_{k+k'})$
   $= G_{n,k+k'}(y_1, \ldots, y_{k+k'}, z_1, \ldots, z_{k+k'})$.
4. *For every $\mathbf{b} \in \overline{\mathbb{F}}^n$ with at most k non-zero components, $\mathbf{b} \in \mathrm{Im}(G_{n,k})$.*

It follows that $G_{n,k}$ hits any polynomial containing a low-support monomial.

▶ **Fact 22** ([53, 20])**.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial that contains a non-zero monomial of support-size at most k, for some $k \in \mathbb{N}$. Then $f(G_{n,k}) \not\equiv 0$.*

The next property follows from Definition 20 and Fact 21 and states that $G_{n,k}$ forms a chain.

▶ **Observation 23.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-zero polynomial and $k \in \mathbb{N}$ such that $f(G_{n,k}) \not\equiv 0$. Then, for every $\ell \geq k$, $f(G_{n,\ell}) \not\equiv 0$.*

Let $\mathcal{C}$ be a circuit class over a field $\mathbb{F}$. Then we define the class,

$$\mathrm{Res}(\mathcal{C}) \triangleq \{\mathrm{Res}_{x_i}(A, B) \mid A, B \in \mathcal{C} \text{ are irreducible and } i \in \mathrm{var}(A) \cup \mathrm{var}(B)\}.$$

We note that $\mathcal{C} \subseteq \mathrm{Res}(\mathcal{C})$ as for any polynomial $f \in \mathcal{C}$, we can write f as $f = \mathrm{Res}_y(P, Q)$, where $P \triangleq (f + 1) \cdot y + 1$ and $Q \triangleq y + 1$ and $y \notin \mathrm{var}(P)$.[5] The following fact is implicit in [55] and [9]. See a proof in [8].

▶ **Fact 24** (Generator for $\sum^{[2]} \prod \mathcal{C}$)**.** *Let $\mathcal{C}$ be a class of arithmetic circuits over a field $\mathbb{F}$ and $\mathcal{G}$ be a generator for the class $\mathrm{Res}(\mathcal{C})$. Then, $H = \mathcal{G} + G_1$ is a generator for $\sum^{[2]} \prod \mathcal{C}$.*

[5] We are assuming that both the polynomials $(f + 1) \cdot y + 1$ and $y + 1$ are also in $\mathcal{C}$, which is true for all natural classes of polynomials.

2.5 Hardness of Representation

▶ **Definition 25.** *Let $k, n, m \in \mathbb{N}$ and let $A_1 \ldots, A_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials. Define $A \triangleq A_1 + \cdots + A_k$. We say that the set $\{A_1, \ldots, A_k\}$ is m-hard, if and only if either $A \equiv 0$ or for every set $J \subseteq [n]$ of size $|J| = m$, the monomial $\prod_{j \in J} x_j$ does not divide A.*

▶ Remark 26. Defining the identically zero polynomial m-hard may seem unnatural or counter-intuitive. However, it is required for technical reasons. In addition, it follows from the definition that if $n < m$ then any set $\{A_1, \ldots, A_k\}$ is m-hard.

The following fact has been used in many works like [53, 5] etc. See a proof in [8].

▶ **Fact 27** (Hardness of representation implies PIT)**.** *Let $m, n, k \in \mathbb{N}$ and $A_1, \ldots, A_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $A \triangleq A_1 + \cdots + A_k \not\equiv 0$. Suppose further that for every subset $I \subseteq [n]$, the set of restricted polynomials $\{A_1|_{\mathbf{x}_I = \mathbf{0}_I}, \ldots, A_k|_{\mathbf{x}_I = \mathbf{0}_I}\}$ is m-hard. Then A contains a non-zero monomial of support-size at most $(m-1)$ and in particular $A(G_{n,m-1}) \not\equiv 0$. Here $G_{n,m-1}$ is the generator given in Definition 20.*

3 ROFs and Multilinear Bounded-Read Arithmetic Formulae

3.1 ROFs and ROPs

▶ **Definition 28** (Read-once formulas, [53])**.** *Let $\mathbb{F}$ be a field and $\mathbf{x} = \{x_1, \ldots, x_n\}$. A read-once formula (in short, ROF) Φ over $\mathbb{F}$ in $\mathbf{x}$-variables is a binary tree whose leaves are labelled with variables in $\mathbf{x}$ and non-leaf nodes are labelled with $+$ and $\times$. Every variable in $\mathbf{x}$ labels at most one leaf of Φ and every node of Φ is associated with a pair $(\alpha, \beta) \in \mathbb{F}^2$. The computation in Φ proceeds as follows: A leaf node of Φ labelled with $x \in \mathbf{x}$ and (α, β) computes $\alpha x + \beta$. A node v labelled with $\circ \in \{+, \times\}$ and (α, β), and having children v_1 and v_2 computes $\alpha(\Phi_{v_1} \circ \Phi_{v_2}) + \beta$, where Φ_{v_i} is the sub-formula of Φ rooted at v_i.*

We say that a polynomial $A \in \mathbb{F}[\mathbf{x}]$ is a *read-once polynomial* (in short, ROP) if it is computed by an ROF. Note that every ROP is multilinear.

3.1.1 Some Useful Properties of ROFs and ROPs

The following fact shows that the class of read-once formulas is closed under factorization and partial derivatives.

▶ **Fact 29** (Lemmas 3.6 and 3.12 of [53])**.** *Let $A \in \mathbb{F}[\mathbf{x}]$ be an ROP, $i \in [n], I \subseteq [n]$, and $\mathbf{a} \in \mathbb{F}^n$. Then, $A|_{\mathbf{x}_I = \mathbf{a}_I}$, $\frac{\partial A}{\partial x_i}$, and factors of A are ROPs.*

Below fact shows that we can make ROPs $\mathbf{0}$-justified by shifting.

▶ **Fact 30** ([53, 41])**.** *Let $n, k \in \mathbb{N}, \mathbb{F}$ be a field, and $A_1, \ldots, A_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be ROPs. Then, we there exists an $\mathbf{a} \in \mathrm{Im}(G_{n,1})$ in such that for every $t \in [k], A_t(\mathbf{x} + \mathbf{a})$ is $\mathbf{0}$-justified.*

The fact below follows from Theorem 3.10 of [52].

▶ **Fact 31.** *A partial derivative of a $\mathbf{0}$-justified ROP is also $\mathbf{0}$-justified.*

The next observation follows from Definition 28.

▶ **Observation 32.** *Let $A, B \in \mathbb{F}[\mathbf{x}]$ be two variable disjoint ROPs. Then, $A \cdot B$ is an ROP.*

3.1.2 Commutator of an ROP

▶ **Fact 33** (Lemma 3.14 of [52]). *Let $A \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an ROP and $i \neq j \in [n]$ such that $\frac{\partial^2 A}{\partial x_i \partial x_j} \not\equiv 0$. Then, there exist variable disjoint ROPs $B(\mathbf{x}), R(\mathbf{x}, y)$ such that $A = R(\mathbf{x}, B(\mathbf{x}))$ and $\Delta_{i,j} A = R(\mathbf{x}, 0) \cdot \frac{\partial^2 A}{\partial x_i \partial x_j}$.*

Facts 29, Observation 15, and Fact 33 imply the following useful result.

▶ **Corollary 34** (Structure of a commutator of an ROP). *Let $A \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an ROP and $i \neq j \in [n]$. Then, $\Delta_{i,j} A$ is a product of ROPs in $\mathbb{F}[x_1, x_2, \ldots, x_n]$.*

3.1.3 The Hardness of Representation Theorem for Sum of ROPs

▶ **Fact 35** (Hardness of representation for sum of k $\mathbf{0}$-justified ROPs, Theorem 6.1 of [53]). *Let $n, k \in \mathbb{N}$ and $A_1, \ldots, A_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be $\mathbf{0}$-justified ROPs. Suppose $n \geq 3k$. Then, for every collection of sets $J_1, \ldots, J_k \subseteq [n]$ and every collection of field elements $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$, the set $\{\alpha_1 \cdot A_1|_{\mathbf{x}_{J_1} = \mathbf{0}_{J_1}}, \ldots, \alpha_k \cdot A_k|_{\mathbf{x}_{J_k} = \mathbf{0}_{J_k}}\}$ is $3k$-hard.*

When $k = 2$, we can, in fact, show that the polynomials are 3-hard rather than 6-hard. See a formal proof in [8].

▶ **Fact 36** (Hardness of representation for sum of two $\mathbf{0}$-justified ROPs). *Let $\mathbb{F}$ be a field and $A, B \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be two $\mathbf{0}$-justified ROPs. Then the set $\{A, B\}$ is 3-hard.*

Using this fact, we give the following useful result used in Section A.

▷ Claim 37. Let $n \geq 3$ be a natural number and $A, B \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be two $\mathbf{0}$-justified ROPs. Suppose there exists a $J \subseteq [n], |J| = 3$ such that for every $j \in J, A|_{x_j=0} = \alpha_j \cdot B|_{x_j=0}$ for some $\alpha_j \in \mathbb{F}$. Then, $A \sim B$.

Proof. Let $j, k \in J$ be distinct. As A, B are $\mathbf{0}$-justified, it follows from Definition 2 that

$$A|_{x_j=0,x_k=0} = \alpha_j \cdot B|_{x_j=0,x_k=0} \not\equiv 0, \quad A|_{x_j=0,x_k=0} = \alpha_k \cdot B|_{x_j=0,x_k=0} \not\equiv 0.$$

These two equations immediately imply that there exists a non-zero $\alpha \in \mathbb{F}$ such that for every $j \in J, \alpha_j = \alpha$. Hence, for every $j \in J, A|_{x_j=0} = \alpha \cdot B|_{x_j=0}$. Now, suppose $A - \alpha \cdot B \not\equiv 0$. As for every $j \in J, A|_{x_j=0} - \alpha \cdot B|_{x_j=0} \equiv 0$, Fact 8 implies that for every $j \in J, x_j$ divides $A - \alpha \cdot B$. Since B is a $\mathbf{0}$-justified ROP, $\alpha \cdot B$ is also a $\mathbf{0}$-justified ROP. Hence, $\prod_{j \in J} x_j$ divides $A - \alpha \cdot B$, which can not happen because of Fact 36. Thus, $A = \alpha \cdot B$. ◁

3.1.4 Obtaining $\mathbf{0}$-Irreducible ROPs

In this section, we give a procedure to convert an irreducible ROP to a $\mathbf{0}$-irreducible ROP (Definition 5). In particular, if $A \in \mathbb{F}[\mathbf{x}]$ is an irreducible n-variate ROP then we compute an assignment $\mathbf{a} \in \mathbb{F}^n$ such that A is $\mathbf{a}$-irreducible. Observation 7 implies that $A(\mathbf{x} + \mathbf{a})$ is a $\mathbf{0}$-irreducible ROP.

It follows from Claim 19 that if all the commutators of a multilinear polynomial $f \in \mathbb{F}[\mathbf{x}]$ are non-zero and if we can efficiently hit all these commutators, i.e., we can efficiently compute an $\mathbf{a} \in \mathbb{F}^n$ such that for every $i \neq j \in \mathrm{var}(f), (\Delta_{i,j} f)(\mathbf{a}) \neq 0$, then using Observation 7, we transform f to a $\mathbf{0}$-irreducible polynomial. It follows from Corollary 17 that for every $i \neq j \in \mathrm{var}(f), \Delta_{i,j} f \not\equiv 0$ if and only if f is irreducible. Thus, only irreducible multilinear polynomials are eligible to be transformed into $\mathbf{0}$-irreducible polynomials. The following fact from [41] would be used in Claim 39.

▶ **Fact 38** (Theorem 4.2 of [41]). *Let $A \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a non-zero ROP and $G_{n,1}$ be the generator given in Definition 20. Then, $A(G_{n,1}) \not\equiv 0$.*

▷ Claim 39 (Converting a set of irreducible ROPs to **0**-irreducible ROPs). Let $n, m \in \mathbb{N}$ and $A_1, \ldots, A_m \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be irreducible ROPs. Then, there exists an assignment $\mathbf{a} \in \mathbb{F}^n$ in the image of $G_{n,1}$ (see Definition 20) such that $A_\ell(\mathbf{x} + \mathbf{a})$ is a **0**-irreducible ROP for every $\ell \in [m]$.

Proof. As $A_1, \ldots, A_m$ are irreducible, Corollary 17 implies that for every $\ell \in [m], i \neq j \in \mathrm{var}(A_\ell), \Delta_{i,j} A_\ell \not\equiv 0$. It follows from Claim 19 that it is sufficient to show that $G_{n,1}$ hits $A_\ell, \Delta_{i,j} A_\ell$ for every $\ell \in [m], i \neq j \in \mathrm{var}(A_\ell)$. Let

$$\Phi(\mathbf{x}) \triangleq \prod_{\ell \in [m]} A_\ell \prod_{i,j \in \mathrm{var}(A_\ell), i \neq j} \Delta_{i,j} A_\ell.$$

As every A_ℓ is irreducible, it is non-zero (recall that every element of $\mathbb{F}$ is reducible). Then, it follows from Corollary 34 that $\Phi(\mathbf{x})$ is a product of non-zero ROPs in $\mathbb{F}[\mathbf{x}]$. Now, Fact 38 and the multiplicative property of generators together imply that $\Phi(G_{n,1}) \not\equiv 0$. Hence, by definition there exists an $\mathbf{a}$ in the image of $G_{n,1}$ such that $\Phi(\mathbf{a}) \neq 0$. Now, Claim 19 and Observation 7 imply that $A_\ell(\mathbf{x} + \mathbf{a})$ is a **0**-irreducible ROP for every $\ell \in [m]$. ◁

3.2 Multilinear Bounded-Read Arithmetic Formulae

▶ **Definition 40** (Read-k formula). *Let $\mathbb{F}$ be a field and $k \in \mathbb{N}$. An arithmetic* read-k formula *F is a binary tree where every leaf node is labelled either by a variable or an element of $\mathbb{F}$; every other node (or internal node) is labelled by either $+$ or $\times$; every edge is labelled by an element of $\mathbb{F}$; and every variable labels at most k leaves of F. Every leaf node of F computes its label. Suppose v is an internal node of F labelled by $\circ \in \{+, \times\}$ such that v_1, v_2 are the children of v, for every $i \in [2]$, v_i computes $F_{v_i} \in \mathbb{F}[\mathbf{x}]$ and the edge between v and v_i is labelled by $\alpha_i \in \mathbb{F}$. Then, v computes the following polynomial $F_v \triangleq \alpha_1 F_{v_1} \circ \alpha_2 F_{v_2}$. Further, if every node of F computes a multilinear polynomial then F is called a multilinear read-k formula.*

An ROF is a special case of a multilinear bounded-read formula. One of the reasons for studying multilinear bounded-read arithmetic formulae is that developing deep understanding of such formulae might give us good insights about the class of multilinear formulae, which is an important class of arithmetic circuits. Deterministic algorithms for blackbox and whitebox PIT for multilinear bounded-read arithmetic formulae were given in [5]. For the rest of this section, let $k \in \mathbb{N}$ be a fixed constant and $\mathcal{C}_k$ be the class of multilinear read-k formulae over a field $\mathbb{F}$. The following result would be used in the proof of Theorem 3.

▶ **Observation 41.** *Let $k \in \mathbb{N}$, and $A, B \in \mathcal{C}_k$ be variable-disjoint. Then, $A \cdot B \in \mathcal{C}_k$.*

The following fact would play a crucial role in the proof of Theorem 3.

▶ **Fact 42** (Implicit in [5]). *Let $k \in \mathbb{N}$, $m \triangleq (8k \cdot (k+1)^2)^k$, and $\widetilde{A}, \widetilde{B}, \widetilde{R} \in \mathcal{C}_k$ compute n-variate polynomials over a field $\mathbb{F}$. Let $\ell \triangleq m + 3k\lceil \log n \rceil$. Then, there exists an assignment $\mathbf{a} \in \mathrm{Im}(G_{n,\ell})$ such that the polynomials $A \triangleq \widetilde{A}(\mathbf{x} + \mathbf{a}), B \triangleq \widetilde{B}(\mathbf{x} + \mathbf{a})$, and $R \triangleq \widetilde{R}(\mathbf{x} + \mathbf{a})$ satisfy Properties 1, 2, and 3 given in Theorem 57.*

Due to space constraint, we have shifted the remaining content to the Appendix. Since the appendix is also space bound, we refer to the full version of this paper given in [8] for the proofs of some results used in the appendix. We direct the reader to Sections A and B of the Appendix. These sections contain the proofs of Theorems 1, 2, and 3.

4 Discussion and Future Work

In this work, we give a polynomial-time blackbox PIT algorithm for the class $\sum^{[2]}\prod$ ROF. We improve upon a result of [39], which gave a whitebox PIT algorithm for the same class. We also took a step forward in solving an open question in [39]. An efficient deterministic PIT algorithm for the class $\sum^{[k]}\bigwedge$ ROF was listed as an open problem in [39]. We give a polynomial-time deterministic blackbox PIT algorithm for $\sum^{[3]}\bigwedge$ ROF. In addition to these two results, we also give a quasi-polynomial-time deterministic blackbox PIT for $\sum^{[3]}\bigwedge\mathcal{C}_k$, where $\mathcal{C}_k$ is the class of multilinear read-k arithmetic formulae over a field $\mathbb{F}$. All our results work over any field. The common thread between these three results is the hardness of representation approach (see Section 2.5). Now we list some open questions.

- PIT for $\sum^{[3]}\prod$ ROF: Our PIT algorithm for $\sum^{[2]}\prod$ ROF crucially depends on the fact that the fan-in of the topmost + gate in the circuits of this class is exactly two. In particular, the resultant based approach used in our algorithm only works in the top fan-in equal to two regime. It is not clear how to lift the resultant-based approach to $\sum^{[3]}\prod$ ROF. Can we come up with some technique that not only yields efficient PIT algorithm for $\sum^{[3]}\prod$ ROF, but also has the potential to extend to PIT for $\sum^{[k]}\prod$ ROF, where k is a constant?
- PIT for $\sum^{[k]}\bigwedge$ ROF: An efficient PIT algorithm for this class would solve an open question given in [39]. Our PIT algorithm for $\sum^{[3]}\bigwedge$ ROF is based on a hardness of representation theorem, which we prove for this class. Can we prove the hardness of representation for $\sum^{[k]}\bigwedge$ ROF? In this direction, we note the following conjecture.

  ▶ **Conjecture 43.** *Let $k, n \in \mathbb{N}$, and $A_1, \ldots, A_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be* **0***-justified ROPs. Then, there exists a monotone function $\varphi : \mathbb{N} \to \mathbb{N}$ such that for any $e_1, \ldots, e_k \in \mathbb{N}$, the set $\{A_1^{e_1}, \ldots, A_k^{e_k}\}$ is $\varphi(k)$-hard (see Definition 25).*

  We remark that this conjecture is true when every $e_i = 1$. In particular, [53] showed that for any constant k, the set $\{A_1, \ldots, A_k\}$ is $3k$-hard (see Fact 35). In addition, for the special case when the A_i-s are products of linear forms over the reals, it was shown in [53], based on a result of [47], that set $\{A_1^{e_1}, \ldots, A_k^{e_k}\}$ is $R_{\mathbb{R}}(k)$-hard (for arbitrary e_i-s) where $R_{\mathbb{R}}(k)$ is the so-called "Rank Bound over the reals". Finally, in [33] it was shown that $R_{\mathbb{R}}(k) = k^{O(k)}$ and improved to $R_{\mathbb{R}}(k) = O(k^2)$ in [49].
- PIT for $\sum^{[2]}\prod\mathcal{C}_k$: The approach used in the proof of Theorem 1 would immediately solve this problem, provided we are able to efficiently compute a common irreducibility preserving assignment (see Definition 5) of a set of multilinear read-k arithmetic formulae. We know that if we could efficiently hit all the commutators of these formulae then such an assignment can be computed efficiently (see Claim 19). In case of ROFs, it turns out that a commutator of an ROF is a product of ROF (see Corollary 34). What can we say about the structure of commutators of multilinear bounded-read arithmetic formulae?

References

1. M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of the 25th FSTTCS*, volume 3821 of *LNCS*, pages 92–105, 2005.
2. M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.
3. M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: Hitting sets, lower bounds for depth-d occur-k formulas and depth-3 transcendence degree-k circuits. *SIAM J. Comput.*, 45(4):1533–1562, 2016.

4 M. Agrawal, C. Saha, and N. Saxena. Quasi-polynomial hitting-set for set-depth-*delta* formulas. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 321–330, 2013.

5 M. Anderson, D. van Melkebeek, and I. Volkovich. Derandomizing polynomial identity testing for multilinear constant-read formulae. *Computational Complexity*, 24(4):695–776, 2015.

6 D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, January 1993.

7 V. Bhargava, S. Saraf, and I. Volkovich. Linear independence, alternants and applications. In *STOC '23: 55th Annual ACM SIGACT Symposium on Theory of Computing, Orlando, Florida, June 20-23, 2023*. ACM, 2023.

8 P. Bisht, N. Gupta, and I. Volkovich. Towards identity testing for sums of products of read-once and multilinear bounded-read formulae. In *Electronic Colloquium on Computational Complexity*, 2023. URL: `https://eccc.weizmann.ac.il/report/2023/109/`.

9 P. Bisht and I. Volkovich. On solving sparse polynomial factorization related problems. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 10:1–10:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.FSTTCS.2022.10`.

10 D. Bshouty and N. H. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *JCSS*, 56(1):112–124, 1998.

11 N. H. Bshouty and R. Cleve. Interpolating arithmetic read-once formulas in parallel. *SIAM J. on Computing*, 27(2):401–413, 1998.

12 N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM J. on Computing*, 24(4):706–735, 1995.

13 N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning boolean read-once formulas with arbitrary symmetric and constant fan-in gates. *JCSS*, 50:521–542, 1995.

14 N.H. Bshouty, T.R. Hancock, and L. Hellerstein. Learning boolean read-once formulas over generalized bases. *J. Comput. Syst. Sci.*, 50(3):521–542, June 1995.

15 D. A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms – An introduction to computational algebraic geometry and commutative algebra (4. ed.)*. Undergraduate texts in mathematics. Springer, 2015.

16 R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

17 P. Dutta, P. Dwivedi, and N. Saxena. Deterministic identity testing paradigms for bounded top-fanin depth-4 circuits. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 11:1–11:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

18 Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2007.

19 M. A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In *FOCS*, 2015.

20 M. A. Forbes, R. Saptharishi, and A. Shpilka. Pseudorandomness for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 867–875, 2014. Full version at `https://eccc.weizmann.ac.il/report/2013/132`. `doi:10.1145/2591796.2591816`.

21 M. A. Forbes and A. Shpilka. Explicit noether normalization for simultaneous conjugation via polynomial identity testing. In *APPROX-RANDOM*, pages 527–542, 2013.

22 J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999.

23 K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Kluwer, 1992.

24 N. Gupta, C. Saha, and B. Thankey. Equivalence test for read-once arithmetic formulas. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 4205–4272. SIAM, 2023.

25 R. Gurjar, A. Korwar, and N. Saxena. Identity testing for constant-width, and commutative, read-once oblivious abps. In *31st Conference on Computational Complexity, CCC*, pages 29:1–29:16, 2016. doi:10.4230/LIPIcs.CCC.2016.29.

26 R. Gurjar, A. Korwar, N. Saxena, and N. Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. In *30th Conference on Computational Complexity, CCC*, pages 323–346, 2015. doi:10.4230/LIPIcs.CCC.2015.323.

27 T. R. Hancock and L. Hellerstein. Learning read-once formulas over fields and extended bases. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory (COLT)*, pages 326–336, 1991.

28 J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*, pages 262–272, 1980.

29 V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2003.

30 M. Karchmer, N. Linial, I. Newman, M. Saks, and A. Wigderson. Combinatorial characterization of read-once formulae. *Discrete Math.*, 114(1–3):275–282, April 1993.

31 Z. S. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich. Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. *SIAM J. on Computing*, 42(6):2114–2131, 2013.

32 Z. S. Karnin and A. Shpilka. Deterministic black box polynomial identity testing of depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity (CCC)*, pages 280–291, 2008.

33 N. Kayal and S. Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2009. Full version at https://eccc.weizmann.ac.il/report/2009/032.

34 N. Kayal and N. Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007.

35 A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.

36 M. Kumar and R. Saptharishi. Hardness-randomness tradeoffs for algebraic computation. *Bulletin of EATCS*, 3(129), 2019.

37 N. Limaye, S. Srinivasan, and S. Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 804–814. IEEE, 2021.

38 L. Lovasz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computing Theory*. Akademia-Verlag, 1979.

39 M. Mahajan, B.V.R. Rao, and K. Sreenivasaiah. Building above read-once polynomials: Identity testing and hardness of representation. *Algorithmica*, 76:890–909, 2016.

40 D. Medini and A. Shpilka. Hitting sets and reconstruction for dense orbits in vp_{e} and ΣΠΣ circuits. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 19:1–19:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

41 D. Minahan and I. Volkovich. Complete derandomization of identity testing and reconstruction of read-once formulas. *TOCT*, 10(3):10:1–10:11, 2018. doi:10.1145/3196836.

42 M. Neunhöffer, 2007. Lecture notes on finite fields - Module MT 5826, Chapter 4, Link - http://www.math.rwth-aachen.de/homes/Max.Neunhoeffer/Teaching/ff/ffchap4.pdf.

43 S. Peleg and A. Shpilka. Polynomial time deterministic identity testing algorithm for $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$ circuits via edelstein-kelly type theorem for quadratic polynomials. 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 259–271. ACM, 2021.

44 C. Ramya and B.V.R. Rao. Lower bounds for sum and sum of products of read-once formulas. *ACM Transactions on Computation Theory (TOCT)*, 11(2):1–27, 2019.

45 C. Saha and B. Thankey. Hitting sets for orbits of circuit classes and polynomial families. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 50:1–50:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

46 S. Saraf and I. Volkovich. Blackbox identity testing for depth-4 multilinear circuits. *Combinatorica*, 38(5):1205–1238, 2018.

47 N. Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 40(1):200–224, 2011.

48 N. Saxena and C. Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012.

49 N. Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33, 2013.

50 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

51 A. Shpilka and I. Volkovich. On the relation between polynomial identity testing and finding variable disjoint factors. In *Automata, Languages and Programming, 37th International Colloquium (ICALP)*, pages 408–419, 2010. Full version at `https://eccc.weizmann.ac.il/report/2010/036`.

52 A. Shpilka and I. Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014.

53 A. Shpilka and I. Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015.

54 A. Sinhababu and T. Thierauf. Factorization of polynomials given by arithmetic branching programs. *computational complexity*, 30(2):1–47, 2021.

55 I. Volkovich. Deterministically factoring sparse polynomials into multilinear factors and sums of univariate polynomials. In *APPROX-RANDOM*, pages 943–958, 2015.

56 I. Volkovich. Characterizing arithmetic read-once formulae. *ACM Transactions on Computation Theory (ToCT)*, 8(1):2, 2016. `doi:10.1145/2858783`.

57 R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226, 1979.

A PIT for $\sum^{[2]} \prod$ ROF

This section is devoted to the proof of Theorem 1, which is based on the proof overview given in Section 1.3.1. We first present some results related to the resultant of two co-prime and **0**-irreducible ROPs in Section A.1. These results are required for proving Theorem 48. Then, using Theorem 48, we give a proof of Theorem 1 in Section A.2.

A.1 Properties of the Resultant of two 0-Irreducible ROPs

We note some important results related to the resultant of two n-variate, co-prime, and **0**-irreducible ROPs A, B. The most important result that we prove is a hardness of representation theorem for $\mathrm{Res}_x(A, B)$ (Lemma 45). A key consequence of this result is that there exists a monomial of support (at most) two in $\mathrm{Res}_x(A, B)$ (Corollary 46). Using this, we show in Lemma 47 that $\mathrm{Res}_x(A, B)(G_{n,3}) \not\equiv 0$, where A, B are any co-prime ROPs (not necessarily **0**-irreducible). We start with the following claim. Its proof is given in [8].

▷ Claim 44. Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be multilinear polynomials, where f is $\mathbf{0}$-irreducible and $i \neq j \in [n]$ such that $\mathrm{var}(f) \setminus \{i, j\} \neq \emptyset$. Suppose x_j divide $\mathrm{Res}_{x_i}(f, g)$. Then, $f|_{x_j=0}$ divides $g|_{x_j=0}$.

The following lemma lies at the heart of the proof of Theorem 1. We show that the resultant of two $\mathbf{0}$-irreducible ROPs is 3-hard (see Definition 25).

▶ **Lemma 45** (Hardness of representation for the resultant of two $\mathbf{0}$-irreducible ROPs)**.** *Let $n \geq 3$ be a natural number and $A, B \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be $\mathbf{0}$-irreducible ROPs. Let $i \in [n]$ be such that $\mathrm{Res}_{x_i}(A, B) \not\equiv 0$. Then $\mathrm{Res}_{x_i}(A, B)$ is 3-hard.*

Proof. Suppose for contradiction that $\mathrm{Res}_{x_i}(A, B)$ is not 3-hard, i.e., $\mathrm{Res}_{x_i}(A, B) \not\equiv 0$ and $\exists J \subseteq [n], |J| = 3$ such that for every $j \in J, x_j$ divides $\mathrm{Res}_{x_i}(A, B)$. We claim that this implies either $J \subseteq \mathrm{var}(A)$ or $J \subseteq \mathrm{var}(B)$. As x_j divides $\mathrm{Res}_{x_i}(A, B)$ for every $j \in J$, we get that $j \in \mathrm{var}(A) \cup \mathrm{var}(B)$. Observe that this implies either $|\mathrm{var}(A) \cap J| \geq 2$ or $|\mathrm{var}(B) \cap J| \geq 2$. Suppose the former is true and $j \neq k \in \mathrm{var}(A) \cap J$. As A is $\mathbf{0}$-irreducible, it follows from Claim 44 that $A|_{x_j=0}$ divides $B|_{x_j=0}$ and $A|_{x_k=0}$ divides $B|_{x_k=0}$. This implies that $j, k \in \mathrm{var}(B)$. Let $\ell \in J \setminus \{j, k\}$. Since $\ell \in \mathrm{var}(A) \cup \mathrm{var}(B)$, we get that either $J \subseteq \mathrm{var}(A)$ or $J \subseteq \mathrm{var}(B)$.

By using the fact that B is also $\mathbf{0}$-irreducible and by using a similar argument as above, it follows that $J \subseteq \mathrm{var}(A) \cap \mathrm{var}(B)$. This implies that for every $j \in J$, there exists a non-zero $\alpha_j \in \mathbb{F}$ such that $A|_{x_j=0} = \alpha_j \cdot B|_{x_j=0}$. Since B is $\mathbf{0}$-irreducible, $|\mathrm{var}(B)| \geq 3$, and $\alpha_j \neq 0$ we get that $\alpha_j \cdot B|_{x_j=0} \not\equiv 0$. Since A and B are $\mathbf{0}$-irreducible ROPs, Claim 6 implies that these are also $\mathbf{0}$-justified. Then, it follows from Claim 37 that there exists an $\alpha \neq 0 \in \mathbb{F}$ such that $A = \alpha \cdot B$. But this means that A, B are not co-prime, thus Fact 10 implies that $\mathrm{Res}_{x_i}(A, B) \equiv 0$, which is a contradiction. Hence, $\mathrm{Res}_{x_i}(A, B)$ is 3-hard. ◀

Using Lemma 45, we can now show that $\mathrm{Res}_{x_i}(A, B)$ has a monomial of support-size at most 2 and hence we can hit $\mathrm{Res}_{x_i}(A, B)$ using $G_{n,2}$. Note that $\mathrm{Res}_{x_i}(A, B)$ by definition does not depend on x_i but we can still consider it as a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. Henceforth, we follow this convention for any $(n-1)$-variate polynomial.

▶ **Corollary 46.** *Let $A, B \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be $\mathbf{0}$-irreducible ROPs such that $\mathrm{Res}_{x_i}(A, B) \not\equiv 0$ for some $i \in [n]$. Then, $\mathrm{Res}_{x_i}(A, B)$ contains a monomial of support-size at most 2. In particular, $(\mathrm{Res}_{x_i}(A, B))(G_{n,2}) \not\equiv 0$.*

Proof. Let $R \triangleq \mathrm{Res}_{x_i}(A, B) \not\equiv 0$. Let J be any subset of $\mathrm{var}(R)$. Note that $i \notin J$. Since A, B are $\mathbf{0}$-irreducible and multilinear, we have that $\deg_{x_i}(P) = \deg_{x_i}(P|_{\mathbf{x}_J=\mathbf{0}_J}) = 1$ for both $P = A, B$. Then by Fact 10,

$$R|_{\mathbf{x}_J=\mathbf{0}_J} = \mathrm{Res}_{x_i}(A, B)|_{\mathbf{x}_J=\mathbf{0}_J} = \mathrm{Res}_{x_i}(A|_{\mathbf{x}_J=\mathbf{0}_J}, B|_{\mathbf{x}_J=\mathbf{0}_J}). \tag{1}$$

Since A, B are $\mathbf{0}$-irreducible, both $A|_{\mathbf{x}_J=\mathbf{0}_J}, B|_{\mathbf{x}_J=\mathbf{0}_J}$ are also $\mathbf{0}$-irreducible. Then by Lemma 45 we deduce that $R|_{\mathbf{x}_J=\mathbf{0}_J}$ is also 3-hard for any $J \subseteq \mathrm{var}(R)$. Now, we can use Fact 27 to show that R has a monomial of support-size at most 2 and thus $R(G_{n,2}) \not\equiv 0$. ◀

In Corollary 46 we showed how to hit the resultant of two $\mathbf{0}$-irreducible ROPs. In the lemma below, we show how to hit resultant of two general ROPs. A proof of this lemma is given in [8]. Recall the definition of the class $\mathrm{Res}(\mathcal{C})$ and Fact 24 from Section 2.

▶ **Lemma 47.** *G_3 is a generator for the class* $\mathrm{Res}(\mathrm{ROF})$.

A.2 Proof of Theorem 1: PIT for $\sum^{[2]}\prod$ ROF

In Section 1.3, we outlined why it suffices to hit the resultants of two ROFs in order to hit the class $\sum^{[2]}\prod$ ROF. For a general class $\mathcal{C}$, it is implicitly shown in previous works like [55, 9] that it suffices to hit the class $\mathrm{Res}(\mathcal{C})$, which we formally stated in Fact 24. In Lemma 47, we have shown that G_3 is a generator for Res(ROF). As a consequence, we get that $G_{n,4} = G_{n,3} + G_{n,1}$ is a generator for any n-variate polynomial in the class $\sum^{[2]}\prod$ ROF.

▶ **Theorem 48** (G_4 hits $\sum^{[2]}\prod$ ROF)**.** *Let $f \in \sum^{[2]}\prod$ ROF be an n-variate polynomial and $G_{n,4}$ be the generator given in Definition 20. Then, $f \equiv 0$ if and only if $f(G_{n,4}) \equiv 0$.*

Proof. In Lemma 47, we showed that G_3 is a generator for the class Res(ROF). Then by Fact 24, $G_4 = G_3 + G_1$ is a generator for the class $\sum^{[2]}\prod$ ROF, that is, given an n-variate polynomial $f \in \sum^{[2]}\prod$ ROF, $f \equiv 0$ if and only if $f(G_{n,4}) \equiv 0$. ◀

Proof of Theorem 1. We are given an n-variate polynomial $f \in \sum^{[2]}\prod$ ROF such that $\deg(f) \leq d$. Then, Theorem 48 implies that $f(G_{n,4}) \equiv 0$ if and only if $f \equiv 0$. Since $f(G_{n,4})$ is an eight-variate polynomial and has degree at most $n \cdot d$, it is not difficult to show that the zeroness of $f(G_{n,4})$ can be tested in $\mathrm{poly}(n, d)$ time. ◀

B PIT for $\sum^{[3]}\bigwedge\mathcal{C}$

This section is devoted to the proofs of Theorems 2 and 3. Here, we prove a more general result which subsumes these two theorems (see Theorem 58). Its proof goes via a hardness of representation result given in Theorem 57. We first discuss some useful definitions and facts.

Let $\mathcal{I}_\ell$ be the ideal in $\mathbb{F}[x_1, x_2, \ldots, x_n]$ generated by $\langle x_1 x_2 \cdots x_\ell \rangle$ for some $\ell \leq n$.

▶ **Definition 49** (polynomial-hat)**.** *For any polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and monomial ideal $\mathcal{I}_\ell$, we can write f as $f = \tilde{f} + \widehat{f}$, where $\tilde{f} \in \mathcal{I}_\ell$ and $\widehat{f} = f \pmod{\mathcal{I}_\ell}$ is the* unique *polynomial obtained from f after going modulo $\mathcal{I}_\ell$.*

▶ Remark 50. Note that the polynomial $\widehat{f}$ may not be unique for general ideals but here for the monomial ideal $\mathcal{I}_\ell$, we define it uniquely by removing all the monomials in f which are divisible by $x_1 \cdots x_\ell$.

The fact below gives a necessary and sufficient condition on the existence of an r-th primitive root of unity in the algebraic closure of a field. See Theorem 8.2 of [42] for a proof.

▶ **Fact 51.** *Let $\overline{\mathbb{F}}$ be the algebraic closure of a field $\mathbb{F}$ and $r \in \mathbb{N}$. Then $\overline{\mathbb{F}}$ contains an r-th primitive root of unity if and only if $r \not| \mathrm{char}(\mathbb{F})$.*

▶ **Observation 52.** *Let $e \in \mathbb{N}$, $\mathbb{F}$ be a field containing an e-th primitive root of unity ω, and x, y be two variables. Then, $x^e - y^e = \prod_{\ell \in [e]}(x - \omega^\ell y)$.*

B.1 Some Useful Results

In this section, we give a set of results required for the proofs of Theorems 2 and 3. The following result generalizes Claim 37. A proof of Claim 53 is given in [8].

▷ Claim 53. Let $n, m \in \mathbb{N}$, $n \geq m \geq 2$ and $A, B \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be two $\mathbf{0}$-justified polynomials such that for every $\beta \in \mathbb{F}$, the set $\{A,\ \beta \cdot B\}$ is m-hard. Suppose there exists a set $J \subseteq [n], |J| = m$ such that for every $j \in J$, there exists an $\alpha_j \in \mathbb{F}$ satisfying $A|_{x_j=0} = \alpha_j \cdot B|_{x_j=0}$. Then, $A \sim B$.

The proofs of the Claims 54, 56, and Lemma 55 are given in the full version [8].

▷ Claim 54. Let $\mathbb{F}$ be a field, $n, e, d \in \mathbb{N}$, such that $2 \leq d \leq e$ and $\mathrm{char}(\mathbb{F})$ does not divide e. Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be multilinear polynomials such that h is non-constant. Suppose $f^e - g^e = h^d$. Then, we get the following.

1. If $d \geq 2$ then $d = e$ and $f \sim g \sim h$.
2. If $d = 1$ then $e = 2$.

Let $n, r \in \mathbb{N}, r \leq n$, $\mathcal{P}_r$ be the monomial $x_1 \ldots x_r$, and $\mathcal{I}_r \triangleq \langle \mathcal{P}_r \rangle$ be the monomial ideal in $\mathbb{F}[x_1, x_2, \ldots, x_n]$. For the lemma below, recall Definition 49. Here we consider a polynomial f w.r.t. ideal $\mathcal{I}_r$ and write $f = \tilde{f} + \widehat{f}$, where $\widehat{f} = f \pmod{\mathcal{I}_r}$.

▶ **Lemma 55.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a non-constant* **0***-justified multilinear polynomial such that $f = g \cdot h + v \cdot \mathcal{P}_r$, where $3 \leq r \leq n$ and $v, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ are arbitrary polynomials (possibly non-multilinear). Then $\widehat{g}$ and $\widehat{h}$ are variable disjoint.*

We shall now use Lemma 55 to prove the following claim that will be used inside the proof of Theorem 57. In this claim, for a polynomial f, we work with the ideal $\mathcal{I}_{m+1}$ and express f as $f = \tilde{f} + \widehat{f}$, where $\widehat{f} = f \pmod{\mathcal{I}_{m+1}}$.

▷ Claim 56. Let $n \geq 3$ be a natural number and let $A, B, R \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be multilinear polynomials that satisfy properties 1, 2 in Theorem 57. Let $H_1 \triangleq A - B$, $H_2 \triangleq A + B$ and $F = H_1 \cdot H_2 - R = v \cdot \mathcal{P}_{m+1}$. For each $i \in \{1, 2\}$, let $J_i = \mathrm{var}(\widehat{H}_i)$ and $I_i = [n] \setminus J_i$. Then $J_1 \cap J_2 = \phi$, $H_1 \sim R|_{\mathbf{x}_{I_1} = \mathbf{0}_{I_1}}$ and $H_2 \sim R|_{\mathbf{x}_{I_2} = \mathbf{0}_{I_2}}$.

B.2 The Hardness of Representation Theorem and PIT

The theorem below is the main technical result of this section.

▶ **Theorem 57** (Hardness of representation for $A^{e_1} - B^{e_2} - R^{e_3}$)**.** *Let $m \geq 2$. Suppose $A, B, R \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ are multilinear polynomials which satisfy the following properties:*

1. *A, B, and R are* **0***-justified.*
2. *For every $J_1, J_2, J_3 \subseteq [n]$ and $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{F}$, the set of polynomials $\{\alpha_1 \cdot A|_{\mathbf{x}_{J_1} = \mathbf{0}_{J_1}},\ \alpha_2 \cdot B|_{\mathbf{x}_{J_2} = \mathbf{0}_{J_2}},\ \alpha_3 \cdot R|_{\mathbf{x}_{J_3} = \mathbf{0}_{J_3}}\}$ is m-hard.*
3. *For any disjoint sets $J_1, J_2 \subseteq [n]$ and for every $\alpha, \beta \in \mathbb{F}$, the set of polynomials $\{\alpha \cdot R|_{\mathbf{x}_{I_1} = \mathbf{0}_{I_1}} \cdot R|_{\mathbf{x}_{I_2} = \mathbf{0}_{I_2}},\ \beta \cdot R\}$ is m-hard, where $I_1 = [n] \setminus J_1$ and $I_2 = [n] \setminus J_2$.*

Let $e_1, e_2, e_3 \in \mathbb{N}$ such that $e_1 \geq e_2 \geq e_3$. Then, the set $\{A^{e_1}, -B^{e_2}, -R^{e_3}\}$ is $(m+1)$-hard.

Proof. Let $F \triangleq A^{e_1} - B^{e_2} - R^{e_3}$. To prove that $\{A^{e_1}, -B^{e_2}, -R^{e_3}\}$ is $(m+1)$-hard, either we have to show that $F \equiv 0$ or for every subset $J \subseteq [n], |J| = m+1$, the monomial $\prod_{j \in J} x_j$ does not divide F. If $F \equiv 0$, there is nothing to prove. If $n < m+1$ then $F \equiv 0$. Therefore, we can assume without loss of generality that $n \geq m+1$ and $F \not\equiv 0$. Assume for the sake of contradiction that there exists a set $J \subseteq [n], |J| = m+1$ such that $\prod_{j \in J} x_j$ divides F. Without loss of generality, let $J = [m+1]$, which implies that the monomial $\mathcal{P}_{m+1} \triangleq x_1 \cdots x_{m+1}$ divides F. In other words, $F \in \mathcal{I}_{m+1}$, where $\mathcal{I}_{m+1}$ is the ideal in $\mathbb{F}[x_1, x_2, \ldots, x_n]$ generated by $\mathcal{P}_{m+1}$. For this proof, we can assume without loss of generality that $\mathbb{F} = \overline{\mathbb{F}}$. This is so because if the monomial $\mathcal{P}_{m+1}$ divides F over the field $\mathbb{F}$ then it also divides it over $\overline{\mathbb{F}}$. As $F \in \mathcal{I}_{m+1}$, there exists a non-zero $v \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that

$$F = v \cdot \mathcal{P}_{m+1}. \tag{2}$$

If $A, B, R \in \mathbb{F}$ then we immediately get a contradiction. So, we assume without loss of generality that A is non-constant (otherwise, B and R are also constants). Now, we analyze the situation in the following cases.

- **Case 1. $e_1 > e_2$**: Let $j \in [m+1]$ be such that $\mathrm{var}(A) \setminus \{j\} \neq \emptyset$. Since $m+1 \geq 2$, such a j always exists. Then, Equation (2) implies $F|_{x_j=0} = (A|_{x_j=0})^{e_1} - (B|_{x_j=0})^{e_2} - (R|_{x_j=0})^{e_3} \equiv 0$. As A is $\mathbf{0}$-justified, non-constant, and $\mathrm{var}(A) \setminus \{j\} \neq \emptyset$, we get that $A|_{x_j=0}$ is a non-constant polynomial. Since $A|_{x_j=0}, B|_{x_j=0}, R|_{x_j=0}$ are multilinear, $A|_{x_j=0}$ is non-constant, and $e_1 > e_2$, $F|_{x_j=0} \not\equiv 0$. This means that F is not divisible by x_j, which contradicts the assumption that $F \in \mathcal{I}_{m+1}$.
- **Case 2. $e_1 = e_2 \geq e_3$**: Fix $e = e_1, d = e_3$. Then,

$$F = A^e - B^e - R^d. \tag{3}$$

For the further discussion, we need an e-th primitive root of unity ω in $\mathbb{F}$. As $\mathbb{F}$ is algebraically closed, Fact 51 tells us that if $p \triangleq \mathrm{char}(\mathbb{F})$ does not divide e then ω is always present in $\mathbb{F}$. If $p = 0$ then ω exists. Suppose p is a prime number and p divides e. Then, there exists an $e' \in \mathbb{N}$ such that $e = e' \cdot p$. As $p = \mathrm{char}(\mathbb{F})$, Equations (2) and (3) imply

$$(A^{e'} - B^{e'})^p - R^d = v \cdot \mathcal{P}_{m+1}. \tag{4}$$

First, suppose that R is non-constant. Clearly, there exists a $j \in [m+1]$ such that $\mathrm{var}(R) \setminus \{j\} \neq \emptyset$. It follows from Equation (4) that $((A|_{x_j=0})^{e'} - (B|_{x_j=0})^{e'})^p = (R|_{x_j=0})^d$. As $\mathrm{var}(R) \setminus \{j\} \neq \emptyset$, $R|_{x_j=0}$ is a non-constant multilinear polynomial. Then, $p|d$.
Now, suppose that $R \in \mathbb{F}$. As $\mathbb{F}$ is algebraically closed, we know that $\alpha \triangleq R^{\frac{1}{p}}$ is present in $\mathbb{F}$. Then, $R^d = (\alpha)^{d \cdot p}$. In this case, without loss of generality, we can replace R with α. This is so because observe that Properties 1, 2, and 3 of A, B, R remain intact if R is constant and we replace it with any other constant. This implies that $\{A^e, -B^e, -R^d\}$ is $(m+1)$-hard if and only if $\{A^e, -B^e, -\alpha^{d \cdot p}\}$ is $(m+1)$-hard.
Thus, in both the cases discussed above i.e., $R \in \mathbb{F}$ and R is non-constant, there exists a $d' \in \mathbb{N}$ such that $d = d' \cdot p$. Then, again using the fact that $p = \mathrm{char}(\mathbb{F})$, it follows from Equation (2) that $F = (F')^p = v \cdot \mathcal{P}_{m+1}$. As $F \in \mathcal{I}_{m+1}$, observe that $F' \in \mathcal{I}_{m+1}$. Thus, we can work with F' instead of F. This argument allows us to assume without loss of generality that p does not divide e. Hence, by Fact 51, we get that an e-th primitive root of unity ω is present in $\mathbb{F}$. Then, on substituting $x = A$ and $y = B$ in $x^e - y^e$ in Observation 52, we get the following useful factorization of $A^e - B^e$.

$$A^e - B^e = \prod_{\ell \in [e]} (A - \omega^\ell B). \tag{5}$$

This factorization would be immensely helpful for further analysis. We first assume that $e = 1$. As $d \leq e = 1$, Equation (3) implies that $F = A - B - \beta R$, where $\beta \in \{0, 1\}$. It follows from Property 2 that the set $\{A, -B, -\beta R\}$ is m-hard. Then Definition 25 implies that F can not be divisible by any multilinear monomial having support m. This contradicts our assumption that $F \in \mathcal{I}_{m+1}$. Henceforth, we assume that $e \geq 2$.

- **Sub-case 2.a. R is a field constant**: Suppose $R \equiv 0$. It follows from Equations (3) and (5) that for every $j \in [n]$, there exists an $\ell_j \in [e]$ such that $A|_{x_j=0} = \omega^{\ell_j} B|_{x_j=0}$. As A is $\mathbf{0}$-justified and $n \geq m+1$, there exists a $J \subseteq [n], |J| = m$ such that for every $j \in J, A|_{x_j=0} \not\equiv 0$. Since $m \geq 2$, Claim 53 implies $A \sim B$. Thus, there exists an $\alpha \in \mathbb{F}$ such that $F = \alpha \cdot A^e$. Observe that $A \in \mathcal{I}_{m+1}$. As $n \geq m+1$, we get from Definition 25 that $\{A\}$ is not $(m+1)$-hard. On the other hand, as $\{A\}$ is m-hard by assumption (see Property 2), observe that it is also $(m+1)$-hard. This is a contradiction.

Now, suppose $R \in \mathbb{F} \setminus \{0\}$. Let $j \in [m+1]$ be such that $\mathrm{var}(A) \setminus \{j\} \neq \emptyset$. It follows from Equations (2) and (3) that $(A|_{x_j=0})^e - (B|_{x_j=0})^e = (R|_{x_j=0})^d$. Since $R \in \mathbb{F} \setminus \{0\}$, it is not difficult to show that $A|_{x_j=0}, B|_{x_j=0} \in \mathbb{F}$. But this can not happen as $\mathrm{var}(A) \setminus \{j\} \neq \emptyset$, A is non-constant and $\mathbf{0}$-justified. Thus, $F|_{x_j=0} \not\equiv 0$, which means that x_j does not divide F and hence F is not in $\mathcal{I}_{m+1}$. This is a contradiction.

- **Sub-case 2.b. $d \geq 2$**: Let $J \subseteq [m+1]$ such that $|J| = m$ and for every $j \in J, \mathrm{var}(R) \setminus \{j\} \neq \emptyset$. Let $j \in J$. Since R is $\mathbf{0}$-justified and non-constant, $R|_{x_j=0}$ is non-constant. Then, Equations (2) and (3) imply $(A|_{x_j=0})^e - (B|_{x_j=0})^e = (R|_{x_j=0})^d$. It follows from Point 1 of Claim 54 that $d = e$ and for every $j \in J$, $A|_{x_j=0} \sim B|_{x_j=0} \sim R|_{x_j=0}$. Since $|J| = m \geq 2$, it is not difficult to see from Claim 53 that $A \sim B \sim R$. Thus, there exists an $\alpha \in \mathbb{F}$ such that $F = \alpha \cdot A^e$. Since, by assumption, $F \in \mathcal{I}_{m+1}$, observe that $A \in \mathcal{I}_{m+1}$. But this can not happen as A is m-hard, which implies that A can not be divisible by any multilinear monomial of support m. Thus, we get a contradiction.
- **Sub-case 2.c. $d = 1$**: From point 2, we get $e = 2$. Then from Equations (2), (3), and (5), we have $F = (A-B)(A+B) - R = v \cdot \mathcal{P}_{m+1}$. Let $H_1 = A - B$ and $H_2 = A + B$. Then, we can write $R = H_1 \cdot H_2 + v' \cdot \mathcal{P}_{m+1}$, where $v' = -v$. Since $n \geq m+1 \geq 3$, by Lemma 55, $\widehat{H}_1, \widehat{H}_2$ are variable disjoint. Moreover Claim 56 shows that for disjoint sets $J_1 = \mathrm{var}(\widehat{H}_1), J_2 = \mathrm{var}(\widehat{H}_2)$, we have $H_1 \sim R|_{\mathbf{x}_{I_1}=\mathbf{0}_{I_1}}$ and $H_2 \sim R|_{\mathbf{x}_{I_2}=\mathbf{0}_{I_2}}$, where $I_1 = [n] \setminus J_1$ and $I_2 = [n] \setminus J_2$. By Point 3, we deduce that the set $\{H_1 H_2, -R\}$ is m-hard and hence also $(m+1)$-hard. This contradicts that $F = H_1 H_2 - R \in \mathcal{I}_{m+1}$. ◀

This result above plays a pivotal role in proving the following theorem. It is not difficult to argue that Theorems 2 and 3 are special details of the following theorem. See [8] for a proof of this theorem and for more details. Recall Definition 20 for this theorem.

▶ **Theorem 58.** *Let $n \in \mathbb{N}$ and let $\widetilde{A}, \widetilde{B}, \widetilde{R} \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be multilinear polynomials. Suppose $F \triangleq \widetilde{A}^{e_1} + \widetilde{B}^{e_2} + \widetilde{R}^{e_3}$, where $e_1, e_2, e_3 \in \mathbb{N}, e_1 \geq e_2 \geq e_3$. Let m be the parameter mentioned in Theorem 57. Let $H : \mathbb{F}^t \to \mathbb{F}^n$ be a generator such that there exist an assignment $\mathbf{a} \in \mathrm{Im}(H)$ for which the polynomials $A \triangleq \widetilde{A}(\mathbf{x}+\mathbf{a}), B \triangleq \widetilde{B}(\mathbf{x}+\mathbf{a})$, and $R \triangleq \widetilde{R}(\mathbf{x}+\mathbf{a})$ satisfy Properties 1, 2, and 3 given in Theorem 57. Then, $F \equiv 0$ if and only if $F(H + G_{n,m}) \equiv 0$.*

Bandwidth of Timed Automata: 3 Classes

Eugene Asarin
Université Paris Cité, CNRS, IRIF, Paris, France

Aldric Degorre
Université Paris Cité, CNRS, IRIF, Paris, France

Cătălin Dima
LACL, Université Paris-Est Créteil, France

Bernardo Jacobo Inclán
Université Paris Cité, CNRS, IRIF, Paris, France

Abstract

Timed languages contain sequences of discrete events ("letters") separated by real-valued delays, they can be recognized by timed automata, and represent behaviors of various real-time systems. The notion of bandwidth of a timed language defined in [16] characterizes the amount of information per time unit, encoded in words of the language observed with some precision ε.

In this paper, we identify three classes of timed automata according to the asymptotics of the bandwidth of their languages with respect to this precision ε: automata are either meager, with an $O(1)$ bandwidth, normal, with a $\Theta\left(\log \frac{1}{\varepsilon}\right)$ bandwidth, or obese, with $\Theta\left(\frac{1}{\varepsilon}\right)$ bandwidth. We define two structural criteria and prove that they partition timed automata into these 3 classes of bandwidth, implying that there are no intermediate asymptotic classes. The classification problem of a timed automaton is PSPACE-complete.

Both criteria are formulated using morphisms from paths of the timed automaton to some finite monoids extending Puri's orbit graphs; the proofs are based on Simon's factorization forest theorem.

2012 ACM Subject Classification Theory of computation → Quantitative automata; Theory of computation → Timed and hybrid models

Keywords and phrases timed automata, information theory, bandwidth, entropy, orbit graphs, factorization forests

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.10

Related Version *Full Version with Proof Details*: `https://arxiv.org/abs/2310.01941`

Funding This work was supported by the ANR project MAVeriQ ANR-CE25-0012 and by the ANR-JST project CyPhAI.

Acknowledgements We thank the anonymous reviewers for their careful reading and valuable comments which helped to improve the paper.

1 Introduction

The study of the growth rate of formal languages was initiated in the 1950s in the pioneering [10], where the growth of the number of words of length n with respect to n in regular languages was analyzed. Even earlier, visionary Shannon [24] related word counts to the capacity of a discrete noiseless channel. In the early '60s, Kolmogorov [17] conceptualized the counting ("combinatorial") approach to the quantity of information, as compared to the probabilistic and the algorithmic ones. According to this approach, the quantity of information conveyed by an element of a finite set S is just $\log_2 \#S$ bits, and in most cases, asymptotics of this amount w.r.t. some size parameter is studied. Nowadays, counting-based asymptotic analysis of formal languages constitutes a background of the theory of codes [7], and of practical protocols [15, Chapter 6] implemented in every hard disk drive and DVD [14].

© Eugene Asarin, Aldric Degorre, Cătălin Dima, and Bernardo Jacobo Inclán;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 10; pp. 10:1–10:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In a nutshell, one can encode a source language with a growth rate A (that is with $\sim 2^{An}$ words of size n) into a channel with growth rate B only if $A \leq B$. The growth rate is also closely related to the entropy rate of subshifts in symbolic dynamics [19].

In the long run, we are porting this approach to timed languages and automata, with both theoretical and practical ambitions. The intrinsic difficulty here is the continuous infinity of such languages and the infinite-state nature of timed automata. For growth rate (or information quantity) per event, we have mostly solved the problem in [3] and [2].

However, in the above, time was seen as data carried by the language, which contrasts with many natural applications, where time describes the actual execution time of the system under study. For this reason, in [16] we formalized the notion of bandwidth, the quantity of information per time unit, given some observation precision, and proved its actual relevance to bounded delay channel coding. Unfortunately, how this bandwidth can be computed for any timed automaton is not yet known. Thus, as a preliminary step, in this paper, we propose a rough qualitative classification of automata into three classes according to the asymptotic of the bandwidth w.r.t. precision. We believe that this qualitative result

- is of interest by itself and provides a relevant classification of qualitative behaviors of timed automata;
- is in line with other classification results for discrete languages;
- involves various and interesting techniques and tools;
- paves the way to the more precise computation of bandwidth.

As a comparison, for multiple classes of languages, the growth rate provides a relevant classification of languages into big (exponential) and small (polynomial) ones, giving important behavioral insights; a gap between polynomial and exponential growth in regular languages is immediate from [10], for context-free languages it was established in [9]. An efficient algorithm for distinguishing between the polynomial and exponential growth of regular and context-free languages is presented in [11]. Finally, for indexed languages, there is no such gap and an intermediate growth is possible [12]. In the same vein but for a slightly different aspect, in [26], nondeterministic finite automata are classified into three classes: finitely, polynomially, and exponentially ambiguous; and the classes are characterized by the presence of specific patterns, as in our study.

For timed regular languages, the growth rate of their volume with respect to the number of events was explored in [3]. In particular, a dichotomy between thin and thick languages was established and related to the Zeno phenomena. In a nutshell, in a thick language (with exponential growth) most timed words are non-Zeno, and have nice properties (they satisfy a pumping lemma, admit discretization, etc.), while in a thin language, all timed words long enough are, in some sense, Zeno and involve very short delays.

In this paper, as already said, we look at the growth rate of timed languages **per time unit**. More precisely, instead of volumes, we study the size of the discretizations of the language, up to precision ε, using ε-nets and ε-separated sets [18]. The logarithms of these quantities are called, resp., ε-entropy and ε-capacity, and the limits of these quantities divided by the time bound of the language are called, resp., entropic and capacitive bandwidth. In [16] we justify the notion of bandwidth proving (up to some details) that a timed source S can be encoded to a timed channel C with bounded delay, iff the bandwidth of S does not exceed the bandwidth of C.

In this article, we partition timed regular languages (and timed automata) into three classes according to the asymptotic behavior of the bandwidth, as ε becomes small. We call them *meager*, with an $O(1)$ (smallest) bandwidth, *normal*, with a $\Theta\left(\log \frac{1}{\varepsilon}\right)$ bandwidth, or *obese*, with $\Theta\left(\frac{1}{\varepsilon}\right)$ (highest) bandwidth. Intuitively, in an obese language, information can be

encoded into time delays with a very high frequency; in a normal language, information can be encoded every few time units; finally, a meager language is extremely constrained and information can be mostly encoded into discrete choices. This trichotomy can be deduced from two structural criteria on timed automata, which exactly characterize the asymptotic classes. Using these structural criteria we also prove that deciding to which class belongs a timed automaton is PSPACE-complete.

Our results build on several ingredients. The bandwidth of a timed language has been defined in [16], itself based on notions of ε-entropy and ε-capacity of functional spaces as studied in [18] and on a pseudo-distance on timed words from [4]. Structural information on timed automata is formulated in terms of orbit graphs from [21], which are discrete objects summarizing reachability relations between states. We had to enrich these graphs with some qualitative information in the form of elements from two finite semirings which represent (1) a rough estimate of the delay between symbols in a timed word and (2) how much choice one may have when taking an edge between two vertices. Structural criteria of meagerness and obesity are formulated in terms of those labels. More precisely, we identify non-meagerness and obesity patterns, and prove that a timed regular language is non-meager resp. obese, whenever it is recognized by a timed automaton with a cycle labeled with an orbit graph containing a corresponding pattern.

The lower bounds take thus the form "whenever a cycle with a pattern is present, the bandwidth is high", and we prove them by providing a direct construction of a big bunch of ε-separated runs along this cycle. The proofs of upper bounds "whenever a cycle with a pattern is absent, the bandwidth is low" are much more involved and rely on Simon's factorization forest theorem, a powerful result relating Ramsey style combinatorics and formal languages theory (see the original article [25] and a modern presentation in [8]). We pinpoint particular properties of Simon factorization whenever one of the non-meagerness or obesity patterns is absent, and build a small ε-net guided by this factorization.

The paper proceeds as follows: in Sect. 2, we recall some notions and tools, and state the problem of classification of timed automata according to their bandwidth. In Sections 3 and 4, we solve the problem for deterministic timed automata by establishing two structural criteria – for very low and very high bandwidth. In Sect. 5, we come up with a classification theorem (and establish the complexity of classification), extend it to the non-deterministic case, and compare our new classes with those from [3]. We conclude with some perspectives. Most technical details and proofs can be found in the ArXiV version (see [5]).

2 Background and Problem Statement

We start this section by recalling our approach [16] to bandwidth (information by time unit) in timed languages and its background: classical ε-entropy and capacity [18] and our pseudo-distance on timed words [4]. Next, we recall timed automata [1] and related constructions, including variants of the region-split form [3] and orbit graphs from [21]. Finally, we present the central notion of the paper: the three classes of timed automata.

2.1 Measuring information in timed languages

Given Σ, a finite alphabet of discrete events, a *timed word* over Σ is an element from $(\Sigma \times \mathbb{R}_+)^*$ of the form $w = (a_1, t_1) \dots (a_n, t_n)$, with $0 \leq t_1 \leq t_2 \cdots \leq t_n$. Number t_i should be interpreted as the *date* at which the event a_i happens. We always put $t_0 = 0$. For any timed word $w = (a_1, t_1) \dots (a_n, t_n)$, we denote its *discrete length* as $|w| \triangleq n$ and its *duration* as $\tau(w) \triangleq t_n$. The *timing* of w is the sequence $t_1 \dots t_n$, and its *untiming* is the sequence $a_1 \dots a_n$. A *timed language* over Σ is a set of timed words over Σ.

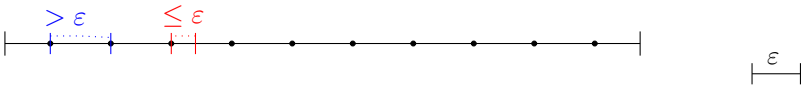


Figure 1 The set of black dots is ε-separated and an ε-net in the segment.

For determining the quantity of information contained in a timed language, we adopt the approach introduced in [18] (for compact functional spaces). Intuitively, when only words that are very close to one another can be observed, then one may not be able to distinguish many different words, thus little information may be conveyed. On the contrary, if there are many words that are set sufficiently far from each other, the fact of choosing among them conveys a large amount of information.

A second point of view is how many words are necessary to approximate the language up to some precision. Both points of view depend on the precision with which we observe, as formalized by the following definition, taken from [18] and adapted to pseudo-metric spaces[1]:

▶ **Definition 1.** *Let (U, d) be a pseudo-metric space and $A \subseteq U$, then:*
- *$M \subseteq A$ is an* ε-separated subset *of A if $\forall x \neq y \in M, d(x, y) > \varepsilon$;*
- *$N \subseteq U$ is an* ε-net *of A if $\forall y \in A, \exists s \in N$ s.t. $d(y, s) \leq \varepsilon$.*

An ε-separated subset of A contains elements of A distant from each other by at least ε, hence distinguishable with precision ε. An ε-net of A allows approximating every element of A with precision ε, see Fig. 1. They give rise to two measures of the information content:

▶ **Definition 2.** *For (U, d) a pseudo-metric space and $A \subseteq U$, the* ε-capacity *$\mathcal{C}_\varepsilon(A)$ and* ε-entropy *$\mathcal{H}_\varepsilon(A)$ are defined as follows[2]:*

$$\mathcal{C}_\varepsilon(A) \triangleq \log \max\{\#M \mid M \ \varepsilon\text{-separated set of } A\}; \ \mathcal{H}_\varepsilon(A) \triangleq \log \min\{\#N \mid N \ \varepsilon\text{-net of } A\}.$$

The following inequality [18] is known to hold for any A subset of U:

$$\mathcal{C}_{2\varepsilon}(A) \leq \mathcal{H}_\varepsilon(A) \leq \mathcal{C}_\varepsilon(A). \tag{1}$$

Two basic cases provide an intuition on the sense of $\mathcal{C}$ and $\mathcal{H}$ as information measures:
- For Σ a finite alphabet (with the discrete metrics), $\mathcal{C}_\varepsilon(\Sigma) = \mathcal{H}_\varepsilon(\Sigma) = \log \#\Sigma$; in words, a letter in Σ brings $\log \#\Sigma$ bits of information.
- Consider now a segment $[0, 1]$, then $\mathcal{C}_\varepsilon[0, 1] \approx \mathcal{H}_{\varepsilon/2}[0, 1] \approx \log 1/\varepsilon$. In words, a real number observed with precision ε contains $\log 1/\varepsilon$ bits of information, intuitively corresponding to $\log 1/\varepsilon$ digits after the binary point.

In the developments thereafter, U will typically be U_Σ, the universal timed language on the working alphabet Σ, while different regular timed languages on Σ will play the role of A. The pseudo-metric will be the one we describe next.

To give a meaningful interpretation of capacity and entropy, an appropriate metric modeling the ability of the observer is required, that is, the ability to distinguish between two words. To this purpose, we imagine an observer that reads the discrete letters of the word exactly (they can determine whether or not a letter has occurred) but with some imprecision

[1] In pseudo-metric spaces one may have $d(x, y) = 0$ for $x \neq y$, but all other axioms of the distance hold. Moreover we also allow infinite distances.
[2] all logarithms in this article are base 2

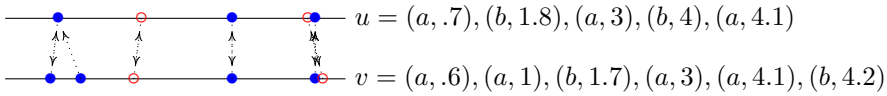


Figure 2 Pseudo-distance between two timed words (dotted lines with arrows represent directed matches between letters). $\overrightarrow{d}(u, v) = 0.2$; $\overrightarrow{d}(v, u) = 0.3$, thus $d(u, v) = 0.3$.

w.r.t. time, so they cannot determine when two letters are very close to one another, which one came before the other, and not even how many times a letter was repeated within a short interval. Hence, we use the following pseudo-distance (similar to the Hausdorff distance between sets):

▶ **Definition 3** ([4]). *The* pseudo-distance d *between two timed words* $w = (a_1, t_1) \dots (a_n, t_n)$ *and* $v = (b_1, s_1) \dots (b_m, s_m)$ *is defined by (with the convention* $\min \emptyset = \infty$*)*

$$\overrightarrow{d}(w, v) \triangleq \max_{i \in \{1..n\}} \min_{j \in \{1..m\}} \{|t_i - s_j| : a_i = b_j\}; \qquad d(w, v) \triangleq \max(\overrightarrow{d}(w, v), \overrightarrow{d}(v, w)).$$

The definition is illustrated on Fig. 2. We remark that $d(w, v) = 0$ whenever w and v only differ by order and quantity of simultaneous letters, e.g. for $w = (a, 1)(b, 1)$ and $v = (b, 1)(b, 1)(a, 1)$. For this reason, d is only a pseudo-distance.

There are two (related) reasons for using this pseudo-distance rather than more straightforward "uniform" distances such as defined in [13]: (1) it provides non-trivial values even when the two words w and v have a different number of events; (2) the set of timed words with duration bounded by some T is compact, which ensures that finite ε-nets always exist.

We now recall the notion of bandwidth, as introduced in [16], a characterization of the ability of a language to produce or convey information *per time unit* in the long run. Bandwidths come in two versions, depending on whether it is about producing or transmitting information. Bandwidths are asymptotics, with respect to time, of the growth rate of ε-capacity or ε-entropy, which finally yield quantities expressed in bits per time unit. We use here the notion of *timed slice* of a language L at some $T \in \mathbb{R}_+$, defined as $L_T \triangleq \{w : w \in L, \tau(w) \leq T\}$.

▶ **Definition 4** ([16]). *The* ε-entropic *and* ε-capacitive bandwidths *of a language* L *are*

$$\mathcal{BH}_\varepsilon(L) \triangleq \limsup_{T \to \infty} \frac{\mathcal{H}_\varepsilon(L_T)}{T}; \qquad \mathcal{BC}_\varepsilon(L) \triangleq \limsup_{T \to \infty} \frac{\mathcal{C}_\varepsilon(L_T)}{T}.$$

The following relationship between the two bandwidths is immediate from (1):

$$\mathcal{BC}_{2\varepsilon}(L) \leq \mathcal{BH}_\varepsilon(L) \leq \mathcal{BC}_\varepsilon(L). \tag{2}$$

2.2 Timed Automata

Regular timed languages are timed languages recognized by timed automata introduced in [1]. In most of this paper, as in [3, 21] we deal with deterministic timed automata (DTA). For a set of variables Ξ, G_Ξ denotes the set of constraints expressible as conjunctions of inequalities of the form $\xi \sim b$ with $\xi \in \Xi$, $\sim \in \{<, \leq, >, \geq\}$ and b an integer ≥ 0.

▶ **Definition 5.** *A* timed automaton *is a tuple* $(Q, X, \Sigma, \Delta, S, I, F)$*, where*

- Q *is the finite set of discrete locations;*
- X *is the finite set of clocks;*

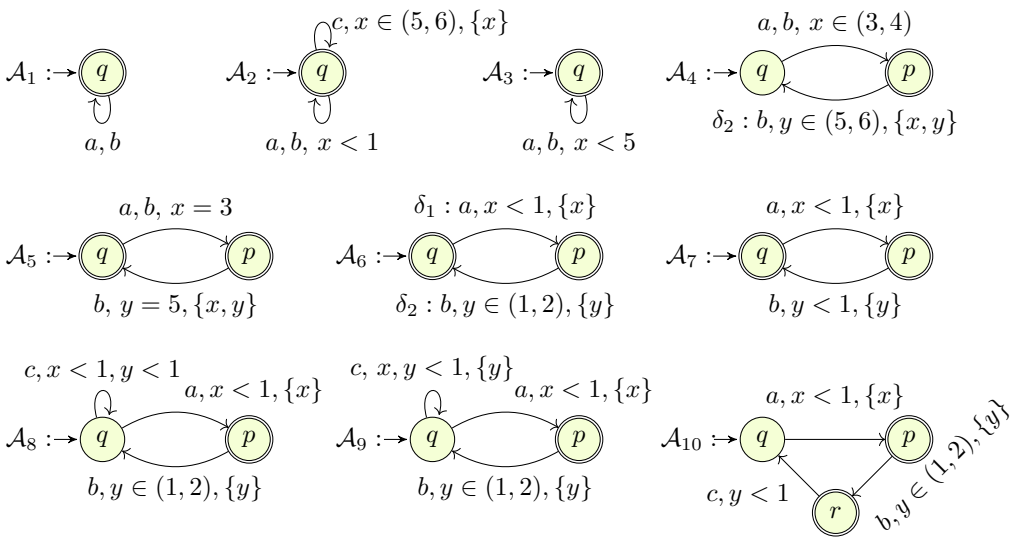


Figure 3 10 running examples of timed automata, $I(q) = \{\mathbf{0}\}$, $F(\cdot) = \mathbf{true}$ for marked locations.

- Σ *is a finite alphabet;*
- $S, I, F : Q \to G_X$ *define respectively the starting, initial, and final clock values for each location;*
- $\Delta \subseteq Q \times Q \times \Sigma \times G_X \times 2^X$ *is the transition relation, whose elements are called edges.*

A timed automaton is deterministic *if* $\{(q, x) | x \models I(q)\}$ *is a singleton and for any two edges* $(q, q_1, a, \mathfrak{g}_1, \mathfrak{r}_1)$ *and* $(q, q_2, a, \mathfrak{g}_2, \mathfrak{r}_2)$ *with* $q_1 \neq q_2$*, the constraint* $\mathfrak{g}_1 \wedge \mathfrak{g}_2$ *is non-satisfiable.*

The role of S, I, F is the following: $S(q)$ denotes the "starting constraint" which must be satisfied by clock values when entering location q. Accepting runs should start in configuration (q, x) with x satisfying $I(q)$ and end by a transition to (q', x') with x' satisfying $F(q')$.

Throughout the paper, we use as running examples the 10 automata described on Fig. 3.

2.2.1 Semantic details and notations

We define two operations that can be applied to any clock vector $x \in [0, \infty)^X$: first, given $t \geq 0$, the *timed successor* of x by t, denoted $x + t \in [0, \infty)^X$, is the clock vector such that for all clocks $c \in X$, $(x + t)_c = x_c + t$; second, given a set of clocks $\mathfrak{r} \subset X$, the $\mathfrak{r}$-*reset of* x is the clock vector $x[\mathfrak{r}] \in [0, \infty)^X$ defined as $x[\mathfrak{r}]_c = 0$ whenever $c \in \mathfrak{r}$ and $x[\mathfrak{r}]_c = x_c$ otherwise.

The following objects will be associated with a timed automaton, and used in the sequel (for all types of sequences, $\circ$ denotes the concatenation and ϵ the empty sequence; the concatenation is lifted in the usual way to sets of sequences, seen as languages):

- For any $\delta \in \Delta$ we define the projections src_δ (source), dst_δ (destination), lbl_δ (label), $\mathfrak{g}_\delta$ (guard) and $\mathfrak{r}_\delta$ (set of reset clocks) such that $\delta = (src_\delta, dst_\delta, lbl_\delta, \mathfrak{g}_\delta, \mathfrak{r}_\delta)$.
- *Edges* are elements of Δ, *edge sequences* are elements of Δ^*, *paths* are edge sequences $\delta_1 \dots \delta_n$ with $dst_{\delta_i} = src_{\delta_{i+1}}$ for all i.
- A *state* is a pair of a discrete location and of a clock vector, i.e. $(q, x) \in Q \times [0, \infty)^X$.
- A *run* is a sequence of the form: $\rho = (q_0, x_0) \overset{(\delta_1, t_1)}{\to} (q_1, x_1) \dots \overset{(\delta_n, t_n)}{\to} (q_n, x_n)$, such that for each i it holds that (by convention $t_0 = 0$)

$$x_i \models S(q_i); src_{\delta_i} = q_{i-1};\ dst_{\delta_i} = q_i;\ x_{i-1} + t_i - t_{i-1} \models \mathfrak{g}_{\delta_i};\ x_i = \mathfrak{r}_{\delta_i}(x_{i-1} + t_i - t_{i-1}). \quad (3)$$

- The following projections from runs to locations, paths, and timed words are defined: $src(\rho) \triangleq (q_0, x_0)$, $dst(\rho) \triangleq (q_n, x_n)$, $path(\rho) \triangleq \delta_1 \dots \delta_n$, $word(\rho) \triangleq (lbl_{\delta_1}, t_1) \dots (lbl_{\delta_n}, t_n)$.
- If ρ_1 and ρ_2 are runs, $\rho_1 \circ \rho_2$ is defined provided: either $\rho_1 = \epsilon$, $\rho_2 = \epsilon$ or $dst(\rho_1) = src(\rho_2)$.
- To any edge sequence π we associate its set of runs $Runs(\pi) \triangleq \{\rho : path(\rho) = \pi\}$. This implies $Runs(\pi) \neq \emptyset$ only when π is a path. Also $Runs(\pi_1 \circ \pi_2) = Runs(\pi_1) \circ Runs(\pi_2)$.
- Given a path π and two clock vectors x, x' we define $Runs(\pi, x, x')$ as the subset of $Runs(\pi)$ starting with clock values x and ending with clock value x'. We denote the timed language $\{word(\rho) | \rho \in Runs(\pi, x, x')\}$ by $L_\pi(x, x')$.
- We also define the language $L_{\bar{\pi}}(x, x')$, where all the guards are replaced by their closures.[3]
- A timed automaton $\mathcal{A}$ defines a timed language $L(\mathcal{A})$: a timed word w belongs to $L(\mathcal{A})$ iff it is in $L_\pi(x, x')$ for some clock vectors x and x' and some path π going from some location q to some location q' with $x \models I(q)$ and $x' \models F(q')$ (so we require also the first state to be initial and the last one to be accepting).

  For a small example consider a 3-edge path $\delta_1\delta_2\delta_1$ in $\mathcal{A}_6$. One of its runs is

  $$(q, 0, 0) \overset{(\delta_1, 0.8)}{\to} (p, 0, 0.8) \overset{(\delta_2, 1.5)}{\to} (q, 0.7, 0) \overset{(\delta_1, 1.7)}{\to} (p, 0, 0.2),$$

and the corresponding timed word is $(a, 0.8), (b, 1.5), (c, 1.7)$.

2.3 Region Abstraction and Region-Splitting

First, let us denote by M the maximal constant appearing in the constraints used in the automaton. Also, for $\xi \in \mathbb{R}$, we denote $\{\xi\}$ its fractional part.

▶ **Definition 6** ([1]). *For a clock vector x, we define $x_\infty = \{c \in X | x_c > M\}$ and $x_\mathbb{N} = \{c \in X | x_c \in \mathbb{N}\}$. Two clock vectors $x, y \in \mathbb{R}_+^X$ are* region-equivalent *whenever*
- *$x_\infty = y_\infty$ and $x_\mathbb{N} \setminus x_\infty = y_\mathbb{N} \setminus y_\infty$;*
- *$\forall c \in X \setminus x_\infty, \lfloor x_c \rfloor = \lfloor y_c \rfloor$;*
- *and $\forall c_1, c_2 \in X \setminus x_\infty$, $\{x_{c_1}\} \leq \{x_{c_2}\}$ iff $\{y_{c_1}\} \leq \{y_{c_2}\}$.*

A region *is an equivalence class of this relation.*

A region R is *M-bounded* whenever $R \subseteq [0, M]^X$. Any M-bounded region R is a simplex of some dimension $d \leq \#X$. We denote by $V(R)$ the set of all $d + 1$ vertices of R.

A folklore result says that each timed automaton can be transformed into a normal form, which can be considered as a (timed) variant of the region graph defined in [1]. The following definition adapts this to our framework while ensuring that starting constraints are bounded:

▶ **Definition 7.** *A region-split M-bounded TA (RsTA) is a DTA $(Q, X, \Sigma, \Delta, S, I, F)$, such that, for any location $q \in Q$:*
- *the starting constraint $S(q)$ defines a non-empty M-bounded region;*
- *all states in $\{q\} \times S(q)$ are reachable from the initial state and co-reachable to a final state;*
- *either $I(q) = S(q)$ is a singleton[4] or $I(q) = \emptyset$;*
- *for any edge $(q, q', a, \mathfrak{g}, \mathfrak{r}) \in \Delta$, it holds that $(\{S(q) + t \mid t \in \mathbb{R}_+\} \cap \mathfrak{g})\,[\mathfrak{r}] = S(q')$.*

▶ **Proposition 8.** *For any DTA, there exists an RsTA accepting the same language, with the same number of clocks, and an exponentially larger number of locations.*

[3] Note that closing guards in an automaton may create new paths but doing so for a single path is "safe".
[4] by definition of DTA this is possible for a unique location q

2.4 Orbits Monoid

Now we define the monoid, based on Puri's orbit graphs [21], and summarizing reachability along paths of an RsTA. We fix some numbering of vertices of regions: for a location q, such that the simplex of the region $S(q)$ has dimension d_q, we have $V(S(q)) = \{v_1^q, \dots, v_{d_q+1}^q\}$.

▶ **Definition 9.** *The finite monoid* $\mathbf{M}$ *of p-orbits contains* $\mathbf{0}, \mathbf{1}$ *and, for any two locations* q *and* q' *all triples* $\langle q, A, q' \rangle$ *with* A *a Boolean* $(d_q + 1) \times (d_{q'} + 1)$*-matrix. All p-orbits with zero matrices are identified with* $\mathbf{0}$*. Composition rules for* $\mathbf{0}$ *and* $\mathbf{1}$ *are standard; and*

$$\langle p, A, p' \rangle \langle q, B, q' \rangle = \begin{cases} \langle p, AB, q' \rangle & \text{if } p' = q; \\ \mathbf{0} & \text{otherwise} \end{cases}$$

(matrices are multiplied using Boolean operations[5]*).*

It is convenient to visualize a p-orbit as a bipartite graph (or an oriented graph whenever $q = q'$), from vertices of $S(q)$ to those of $S(q')$, with edges corresponding to 1s in the matrix.

Following [21][6] we associate a p-orbit to each edge sequence.

▶ **Definition 10** (The p-orbit of a path summarizes reachability)**.** *Given an RsTA, we define a function* γ *from edge sequences to* $\mathbf{M}$*:*

- $\gamma(\epsilon) = \mathbf{1}$ *and whenever sequence* π *is not a path,* $\gamma(\pi) = \mathbf{0}$*;*
- *for* π *a non-empty path from* q *to* q' *let* $\gamma(\pi) = \langle q, A, q' \rangle$*, with* $A_{ij} = 1$ *iff* $L_{\bar{\pi}}(v_i^q, v_j^{q'}) \neq \emptyset$*.*

In graph terms, there is an edge in the orbit graph of π from a vertex v to v' iff (q', v') is reachable from (q, v) along (the closure of) π.

▶ **Proposition 11** (probably [3])**.** *γ is a monoid morphism.*

Thanks to this property, we can easily compute $\gamma(\pi)$: for one edge using the definition, for a longer path by composition: $\gamma(\delta_1 \dots \delta_k) = \gamma(\delta_1) \dots \gamma(\delta_k)$.

It follows from the theory developed in [21], that p-orbits are an efficient abstraction for reachability: for each path π, its p-orbit $\gamma(\pi)$ fully describes the reachability relation between a clock vector x in the starting region of π and a clock vector x' in its ending region, through a run along π. Intuitively, the reason is that any run from (q, x) to (q', x') can be expressed as a convex combination of runs of surrounding vertices of $S(q)$ to vertices of $S(q')$, summarized in the orbit $\gamma(\pi)$.

Consider the running example $\mathcal{A}_6$, its RsTA has its main cycle[7] between q and p with $S(q) = (0, 1) \times \{0\}, S(p) = \{0\} \times (0, 1)$. The p-orbits of its edges are the two graphs on the left of Fig. 4, and their matrix form is:

$$\gamma(q \to p) = \gamma_1 = \left\langle q, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, p \right\rangle; \ \gamma(p \to q) = \gamma_2 = \left\langle p, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, q \right\rangle.$$

The orbits of the two cycles, i.e. the two graphs on the right of Fig. 4, are the following:

$$\gamma(q \to p \to q) = \gamma_1 \cdot \gamma_2 = \left\langle q, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, q \right\rangle; \ \gamma(p \to q \to p) = \gamma_2 \cdot \gamma_1 = \left\langle p, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, p \right\rangle.$$

---

[5] that is $(AB)_{ij} = \bigvee_k (A_{ik} \wedge B_{kj})$

[6] in fact, [21] only considers cyclic paths in the region automaton

[7] there are two more transient locations, not considered here

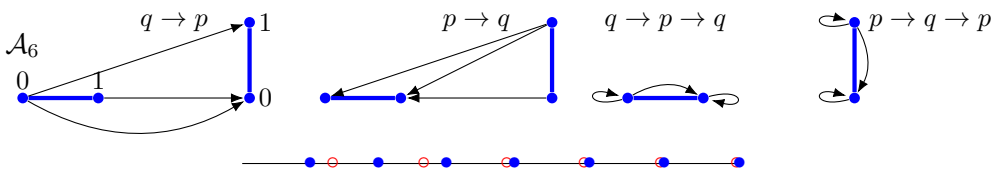


Figure 4 P-orbits for $\mathcal{A}_6$ and a typical word in $L(\mathcal{A}_6)$:
$(a, .9)(b, 1.2)(a, 1.8)(b, 2.4)(a, 2.7)(b, 3.5)(a, 3.6)(b, 4.52)(a, 4.59)(b, 5.53)(a, 5.58)(b, 6.535)(a, 6.575)$.

2.5 Central notion of the paper: the three classes

▶ **Definition 12.** *A timed language L is*
- meager *whenever* $\mathcal{BH}_\varepsilon(L) = O(1)$*;*
- normal *whenever* $\mathcal{BH}_\varepsilon(L) = \Theta(\log \frac{1}{\varepsilon})$*;*
- obese *whenever* $\mathcal{BH}_\varepsilon(L) = \Theta(\frac{1}{\varepsilon})$*,*

as $\varepsilon \to 0$.

It follows immediately from (2) that a similar and equivalent characterization can be done in terms of capacity by replacing $\mathcal{BH}$ by $\mathcal{BC}$. We will now give some intuition concerning the three classes, using running examples.

Automata with obese language: information is produced with a high frequency. Thus, in $\mathcal{A}_1$ every ε seconds one can produce an a, a b, both or nothing, which corresponds to two bits of information, and yields a bandwidth of $2/\varepsilon$ bit/sec. In $\mathcal{A}_2$ such a high-frequency encoding is possible during 1 second, after that one should wait 4 seconds and take the c loop to reset x. In total, the bandwidth will be $\frac{2}{5\varepsilon}$.

Automata with normal language: during one unit of time they typically make $O(1)$ discrete choices, and $O(1)$ real-valued choices. The former convey $O(1)$ bit/sec, the latter $O(\log 1/\varepsilon)$. Thus $\mathcal{A}_4$ makes a cycle in 5 to 6 seconds, with two real-valued choices (duration of stay in p and in q), and the bandwidth is $\frac{2}{5} \log 1/\varepsilon$.

Automata with meager language: In the long run such automata should not allow real-valued choices. $\mathcal{A}_5$ has only discrete choices since its transitions happen at discrete dates only. $\mathcal{A}_3$ can produce a huge amount of information ($2/\varepsilon$ bits every second), but only during the first five seconds of its life. Its bandwidth is $\lim_{T\to\infty} 2/\varepsilon T = 0$. The automaton $\mathcal{A}_6$ is much less evident. It requires a and b to interleave, with a being < 1 second apart and b on the contrary > 1 second apart, as shown on Fig. 4. Such interleaving is possible for any duration T, but real-valued intervals between symbols become more and more constrained. We prove later that this language is indeed meager.

In the rest of the paper, we propose a structural and decidable characterization of DTA establishing membership of their languages into one of these 3 classes, also proving that no other possibility exists.

3 Distinguishing between Meager and Other Automata

For the next two sections, our plan is to characterize the bandwidth of the timed language generated by a cycle looping over a region and to extend this analysis to the whole language. As bandwidth is about choice per unit of time, we enrich the monoid of p-orbits with one of the following two types of information: (1) the amount of choice one may have when taking an edge between two vertices, or (2) a rough estimate of the interval of durations of timed words generated by the cycle, starting in one vertex and ending in another vertex. The first

type of information is useful for discriminating meager from non-meager languages, while the second type will distinguish obese from non-obese languages. We start with the former, slightly simpler to formulate.

3.1 Orbits with Abstracted Freedom

The structural criterion for meagerness is based on an enrichment of the monoid of p-orbits in which matrix entries summarize the dimension of the reachability polytope between vertices. In more detail, matrix entries belong to $\Lambda_f \triangleq \{0, \mathbf{narrow}, \mathbf{wide}\}$, having a semiring structure[8], defined in Table 1. The intuition is the following. Elements of Λ_f summarize reachability polytopes between region vertices through some path: no reachability for 0 (empty polytope), reachability without continuous choice for **narrow** (singleton polytope), presence of at least one continuous choice for **wide** (polytope of dimension ≥ 1). The addition of Λ_f represents the convex union of two compatible sets of runs, while the multiplication corresponds to the concatenation of sets of runs, in particular, two distinct singletons logically add up to a one-dimensional polytope, while the dimension of two concatenated sets of runs is the sum of the dimensions. Hence the monoid describes the dimension of the reachability polytope of a path: its degree of *freedom*.

Table 1 Addition and multiplication of the semiring Λ_f.

$+$	0	**narrow**	**wide**
0	0	**narrow**	**wide**
narrow	**narrow**	**wide**	**wide**
wide	**wide**	**wide**	**wide**

$\times$	0	**narrow**	**wide**
0	0	0	0
narrow	0	**narrow**	**wide**
wide	0	**wide**	**wide**

▶ **Proposition 13.** $(\Lambda_f, +, \times, 0, \mathbf{narrow})$ *is a commutative semiring.*

Thus the definition of the monoid $\mathbf{M}_f$ of f-orbits is identical with Def. 9, with the only difference, that matrix entries belong to the semi-ring Λ_f. Graphically, we represent all non-0 elements of Λ_f by labeled (or colored) edges, and 0 by an absence of edge, see Fig. 5.

▶ **Definition 14** (The f-orbit of a path abstracts its freedom)**.** *Given an RsTA, we define a function γ_f from edge sequences to $\mathbf{M}_f$.*

- $\gamma_f(\epsilon) = \mathbf{1}$. *Whenever sequence π is not a path, $\gamma_f(\pi) = \mathbf{0}$.*
- *For π a non-empty path from q to q', let $\gamma_f(\pi) = \langle q, B, q' \rangle$, with the following matrix entries: for vertices $v = v_i^q \in V(S(q))$ and $v' = v_j^{q'} \in V(S(q'))$*
  - $B_{ij} = \mathbf{0}$ *iff* $L_{\bar{\pi}}(v, v') = \emptyset$;
  - $B_{ij} = \mathbf{narrow}$ *iff* $L_{\bar{\pi}}(v, v')$ *is a singleton;*
  - $B_{ij} = \mathbf{wide}$ *otherwise.*

We remark that in the last case $L_{\bar{\pi}}(v, v')$ has a dimension ≥ 1.

▶ **Proposition 15.** γ_f *is a monoid morphism.*

Again, this property allows compositional computation of $\gamma_f(\pi)$.[9]

---

[8] also known as super-Boolean semiring $\mathbb{SB}$, [22]

[9] Def. 14 may not seem effective but, for a single edge δ, $\gamma_f(\delta)$ can be computed in polynomial time in $\#X$, by solving the constraints of δ, while $\gamma_f(\cdot)$ for longer paths can be computed using the multiplication.

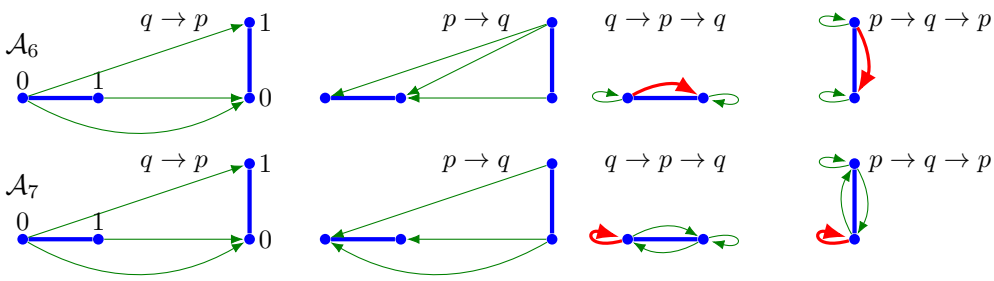


Figure 5 F-orbits for $\mathcal{A}_6$ and $\mathcal{A}_7$; green thin arrows are **narrow**, red thick ones **wide**.

▶ **Definition 16.** *A cycle of an RsTA is* structurally meager *if its f-orbit has no* **wide** *self-loop (in matrix terms, a* **wide** *on the main diagonal). An RsTA is* structurally meager *when all its cycles are structurally meager.*

Consider the running example $\mathcal{A}_6$. The top row of Fig. 5 (which is a colorful version of Fig. 4) illustrates f-orbits $\gamma_f(q \to p) = \gamma'_1$ and $\gamma_f(p \to q) = \gamma'_2$, in matrix form:

$$\left\langle q, \begin{pmatrix} \textbf{narrow} & \textbf{narrow} \\ \textbf{narrow} & 0 \end{pmatrix}, p \right\rangle; \text{ and } \left\langle p, \begin{pmatrix} 0 & \textbf{narrow} \\ \textbf{narrow} & \textbf{narrow} \end{pmatrix}, q \right\rangle.$$

and the f-orbits of the cycles $q \to p \to q$ and $p \to q \to p$, i.e. the products $\gamma'_1\gamma'_2$ and $\gamma'_2\gamma'_1$:

$$\left\langle q, \begin{pmatrix} \textbf{narrow} & \textbf{wide} \\ 0 & \textbf{narrow} \end{pmatrix}, q \right\rangle; \text{ and } \left\langle p, \begin{pmatrix} \textbf{narrow} & 0 \\ \textbf{wide} & \textbf{narrow} \end{pmatrix}, p \right\rangle.$$

Similarly, the bottom row of Fig. 5 presents f-orbits of $\mathcal{A}_7$, and one can observe that $\mathcal{A}_6$ is structurally meager (no **wide** self-loops), while $\mathcal{A}_7$ is not.

The decision procedure for structural meagerness uses the following "pumping lemma":

▶ **Lemma 17** (adapted from [23, Lem. 8.7.1])**.** *Given a finite alphabet Γ, a finite monoid $\mathbf{M}$, and a morphism $\beta : \Gamma^* \longrightarrow \mathbf{M}$, for any $e \in \mathbf{M}$ having $\beta^{-1}(e) \neq \emptyset$, there exists a word $w \in \beta^{-1}(e)$ of a length $\leq \#\mathbf{M}$.*

▶ **Theorem 1.** *Structural meagerness of an RsTA of description size n with m clocks is decidable in* $\mathsf{SPACE}(\log^2(n)\mathsf{poly}(m))$.

Here poly stands for a polynomial. We will also use exp for an exponential function below.

Proof sketch. The RsTA has at most n locations, each orbit can be represented as a couple of locations and an (at most) $(m+1) \times (m+1)$-matrix with elements in Λ_f (where $\#\Lambda_f = 3$). Thus $\#\mathbf{M}_f \leq n^2 \cdot 3^{(m+1)^2} = \mathsf{poly}(n)\mathsf{exp}(m)$. An automaton is structurally meager whenever it contains no cyclic path π with orbit $\gamma_f(\pi)$ having a **wide** entry on the matrix diagonal. By Lem. 17, only cycles up to length $\#\mathbf{M}_f$ need to be checked. We use now the reachability method [20, Sect. 7.3] to detect such a cycle in a space-efficient way, see Listing 1. An auxiliary function `isAPathOrbit(e, h)`, true iff there is a path $\pi \in \gamma_f^{-1}(e)$ of length $\leq 2^h$ in the RsTA, is computed by a divide-and-conquer algorithm. The algorithm requires a call stack of depth $\log \#\mathbf{M}_f$, the size of each stack frame is $O(\log n) + \mathsf{poly}(m)$, thus the computation takes space $(O(\log n) + \mathsf{poly}(m)) \cdot \log \#\mathbf{M}_f = \log^2(n)\mathsf{poly}(m)$. ◀

Listing 1 Deciding structural meagerness.

```
Boolean function isAPathOrbit(e, h)
    if h = 0 return (e=1 || ∃δ : γ_f(δ) =e)
    for all orbits e1, e2
        if e1·e2 = e && isAPathOrbit(e1, h-1) && isAPathOrbit(e2, h-1)
            return true
    return false
 Boolean function isStrucMeager()
    for all cyclic orbits e
        if (containsWideOnDiag(e) && isAPathOrbit(e, log #M_f))
            return false
    return true
```

3.2 Structural Meagerness ⇔ Meagerness

The upper bound can be formulated as follows.

▶ **Proposition 18.** *The timed language of a structurally meager RsTA is meager.*

The proof (see [5]) is done in several steps, two of them using Simon's theorem. Let $\mathcal{A}$ be structurally meager. First, we explore the languages $L((q,x),(q,x))$, of words, looping from a state to itself, and observe that, for any structurally meager cycle π, $L_\pi(x,x)$ is a singleton. Intuitively, words of $L((q,x),(q,x))$ do only discrete choices, and thus convey only $O(1)$ bit/second of information.

Next, we make a finite partition of all the regions into small (ε-sized) cells and define a set $Still_\varepsilon$ of words accepted along a cycle from such a cell to the same cell. Such words are close to singletons and thus convey $O(1)$ bit/second of information when observed with precision ε. On the other hand, visits to different cells by the same cycle (and even by different cycles with the same orbit) always respect a certain partial order $\succeq$ over cells.

Last, we use Puri's reachability characterization [21] and our technique of linear Lyapunov functions [3], together with Simon's theorem to factorize any word in $L_T(\mathcal{A})$ in factors that are either one letter or belong to $Still_\varepsilon^*$. The total number of factors does not depend on T, but only on ε. This factorization makes it possible to build a small ε-net and conclude.

On the other hand, we need a lower bound.

▶ **Proposition 19.** *If an RsTA is not structurally meager then its bandwidth is* $\Omega(\log(1/\varepsilon))$.

The proof idea (see [5]) is to take a cycle that is not structurally meager, and show that it alone provides a bandwidth of $\log(1/\varepsilon)$. The main theorem of this section is now immediate from Props. 18 and 19:

▶ **Theorem 2.** *The language of an RsTA is meager iff the RsTA is structurally meager.*

4 Distinguishing between Obese and Other Automata

4.1 Orbits with Abstracted Duration

The structural criterion for obesity is based on another enrichment of p-orbits, in which the entries of the matrix A belong to a finite semiring $\Lambda_d \triangleq \{\mathbf{instant}, \mathbf{fast}, \mathbf{slow}, 0\}$, with operations detailed in Table 2. Intuitively, the elements of Λ_d represent 4 abstracted classes of durations for sets of runs having the same path, same source, and same destination vertices; similarly to Λ_f, the addition of Λ_d represents the convex union of two compatible sets of runs, while the multiplication corresponds to the concatenation of sets of runs.

Table 2 Addition and multiplication of the semiring Λ_d.

$+$	**instant**	**fast**	**slow**	0
instant	**instant**	**fast**	**fast**	**instant**
fast	**fast**	**fast**	**fast**	**fast**
slow	**fast**	**fast**	**slow**	**slow**
0	**instant**	**fast**	**slow**	0

$\times$	**instant**	**fast**	**slow**	0
instant	**instant**	**fast**	**slow**	0
fast	**fast**	**fast**	**slow**	0
slow	**slow**	**slow**	**slow**	0
0	0	0	0	0

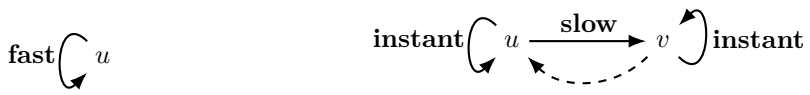


Figure 6 Obesity patterns. Left: Type I. Right: Type II.

▶ **Proposition 20.** $(\Lambda_d, +, \times, 0, \mathbf{instant})$ *is a commutative semiring.*

Similarly to $\mathbf{M}_f$ in the meager case, we define $\mathbf{M}_d$, the monoid of d-orbits, where the only difference is that matrix entries belong to Λ_d instead of Λ_f. As in previous cases, we represent elements of $\mathbf{M}_d$ as labeled graphs.

▶ **Definition 21** (The d-orbit of a path abstracts its speed)**.** *Given an RsTA, we define a function γ_d from edge sequences to $\mathbf{M}_d$.*

- $\gamma_d(\epsilon) = \mathbf{1}$. *Whenever sequence π is not a path, $\gamma_d(\pi) = \mathbf{0}$.*
- *For π a non-empty path from q to q' let $\gamma_d(\pi) = \langle q, B, q' \rangle$, with the following matrix entries: for vertices $v = v_i^q \in V(S(q))$ and $v' = v_j^{q'} \in V(S(q'))$*
  - $B_{ij} = \mathbf{0}$ *iff* $L_{\bar{\pi}}(v, v') = \emptyset$;
  - $B_{ij} = \mathbf{instant}$ *iff $L_{\bar{\pi}}(v, v')$ is a singleton $\{w\}$, with $\tau(w) = 0$;*
  - $B_{ij} = \mathbf{slow}$ *iff $L_{\bar{\pi}}(v, v') \neq \emptyset$ and $\tau(L_{\bar{\pi}}(v, v')) \subseteq [1, \infty)$;*
  - $B_{ij} = \mathbf{fast}$ *otherwise.*

It is not difficult to see that in the last case $\tau(L_{\bar{\pi}}(v, v')) \supseteq [0, 1]$.

▶ **Proposition 22.** *γ_d is a monoid morphism.*

Again, this property allows a compositional computation of $\gamma_d(\pi)$.

4.2 The Obesity Pattern

▶ **Definition 23.** *A cycle in an RsTA is* structurally obese *iff its d-orbit has the obesity pattern of one of the two types presented on Fig. 6, in the sense that either (a) $\gamma_d(\pi)$ has a* **fast** *diagonal element, or (b), for some indices u and v, it has two* **instant** *diagonal elements at (u, u) and (v, v), the element at position (u, v) is* **slow** *and some other realizable d-orbit on the same region has a non-*0 *element at position (v, u).*

An RsTA is structurally obese if *it has a structurally obese cycle.*

Some comments are in order. The above definitions capture the intuition that obese behaviors are "fast" behaviors (cycles of unbounded frequency) that can be repeated during an unbounded total duration, and this can happen in two ways only:

Type I structural obesity: there exists a fast cycle that can be repeated with unbounded frequency for an unbounded accumulated duration.

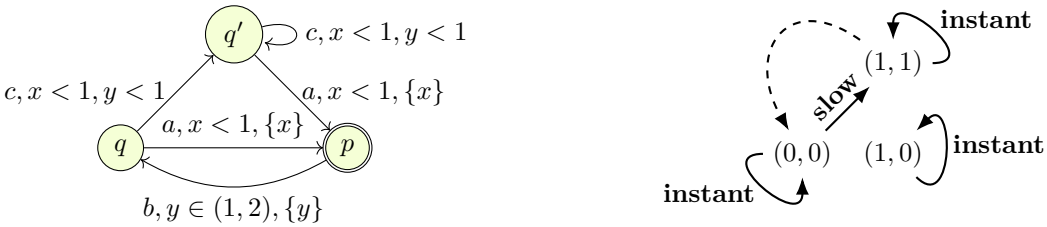


Figure 7 Type II obesity for $\mathcal{A}_8$. Left: its region-split form (transient locations omitted). Right: d-orbit for the self-loop on q' in the simplex $S(q')$; each tuple represents the value for clocks x and y respectively. The "resetting" cycle is $q' \to p \to q \to q'$.

Type II structural obesity: there exists a "Zeno" cycle π (with a d-orbit having a **slow** edge from a vertex u to another vertex v, and **instant** self-loops around both vertices) intersecting with another "resetting" cycle π' (with a p-orbit having an edge from v to u). The cycle π can be repeated with unbounded frequency for a total duration of ≤ 1 time unit, until π' is executed and resets this total, allowing, infinitely often, to spend another time unit in π.

Let us illustrate this definition on examples from Fig. 3. Obesity patterns for $\mathcal{A}_1$ and $\mathcal{A}_2$ can be easily seen. Consider now $\mathcal{A}_8$. The d-orbit of the self-loop above its state q' is drawn in Fig. 7 (right). The type II obesity pattern occurs because, by taking the path $q' \to p \to q \to q'$, one may return from vertex $(1, 1)$ to $(0, 0)$ of the region $S(q')$. This orbit has an edge from $(1, 1)$ to $(0, 0)$, symbolized by the dashed line edge in Fig. 7 (right). Finally, the reader can check that $\mathcal{A}_9$ is not structurally obese.

Similarly to Thm. 1, structural obesity is in PSPACE.

▶ **Theorem 3.** *Structural obesity of an RsTA of description size n with m clocks is decidable in* $\mathsf{SPACE}(\log^2(n)\mathsf{poly}(m))$.

4.3 Structural Obesity ⇔ Obesity

Let us state first the upper bound for non structurally obese automata.

▶ **Proposition 24.** *If an RsTA $\mathcal{A}$ is not structurally obese then its bandwidth is $O(\log(1/\varepsilon))$.*

The proof (see [5]) is quite technical. At its first stage, we carefully explore Zeno cycles, which can convey a lot of information, but only for a short lapse of time. In the second stage we use Simon's theorem on factorization forests [25], and factorize words of duration T in the language, the number of factors is $\sim T$, some of them are Zeno (their total duration is only $O(1)$), others contain only one event or have duration 0. Finally, we make a profit from the factorization and build an ε-net for $L_T(A)$ of the required size.

The lower bound is formulated as follows.

▶ **Proposition 25.** *The language of a structurally obese RsTA is obese.*

The principle of the proof (see [5]) is simple: we identify a cycle with an obesity pattern, and show that it may be used to transmit information with high frequency (every ε sec) which yields the required large bandwidth. When the clocks become too high, to continue looping through the cycle, a "resetting" path is taken.

The main result of this section is now immediate.

▶ **Theorem 4.** *The language of an RsTA is obese iff the automaton is structurally obese.*

5 Gathering Stones: Classification

5.1 Classifying DTA According to Bandwidth Asymptotics

▶ **Theorem 5.** *A timed language accepted by a DTA is either meager, normal, or obese. Given a DTA, the three problems "is it meager?", "is it normal?", "is it obese?" are* PSPACE*-complete.*

Proof. First, we put the DTA into RsTA form, with as many clocks but exponentially more states, as ensured by Prop. 8. If the RsTA is structurally meager (resp. structurally obese), then it is meager (obese) by Thms. 2 and 4. If it is neither structurally meager nor structurally obese, then by Props. 19 and 24 its bandwidth is both $O(\log(1/\varepsilon))$ and $\Omega(\log(1/\varepsilon))$, hence it is normal. PSPACE membership follows from Thms. 1 and 3. PSPACE-hardness can be proved (see [5]) by reduction of reachability [1]. ◀

▶ **Corollary 26.** *The same classification and complexity results also hold for nondeterministic timed automata without ϵ-transitions.*

5.2 Comparing with Thin-Thick Classification

Previously in [3], we established a dichotomy between thin (whose language for n events is subexponentially vanishing as n goes to ∞) and thick automata (whose language has volumes growing exponentially) and proposed a structural characterization according to orbits: thick automata are those which contain a reachable *forgetful* cycle, i.e. a cycle with a p-orbit having a complete graph.

It is natural to compare the approach from [3] with the new one. First, their goals differ: the former assesses how much a time language grows at every discrete event, and the latter handles growth per unit of time. Second, in [3], only bounded guards were considered, because paths through such guards would have infinite volumes. Third, for a similar reason, paths with punctual transitions were ignored, because of their empty volumes, often entailing a thin verdict for trivial reasons. In the new approach, paths with punctual guards can contribute to meager, normal, and even obese, behaviors, while the presence of unbounded guards in a path is not a determining factor with respect to bandwidth class (efficient coding strategies tend to favor as short delays as possible). Nonetheless, in the bounded and non-punctual case, it is possible to examine the same automaton under both approaches and obtain the following result:

▶ **Theorem 6.** *An RsTA with only bounded non-punctual guards cannot be both thick and meager. Also, it cannot be thin and obese of Type I. All other combinations are possible.*

Proof sketch. Consider a thick automaton, with its forgetful cycle π. If its orbit has a single vertex, since π has a non-punctual transition it can be realized by several runs, hence the only edge of $\gamma_f(\pi)$ is **wide**. Otherwise, $\gamma(\pi)$ has a complete SCC of size at least 2, and then all its vertices have **wide** self-loops in $\gamma_f(\pi^2)$.

Now consider an automaton with a structurally obese cycle π of Type I. By definition, $\gamma_d(\pi)$ has a **fast** self-loop. It can be shown that such a cycle must have a complete p-orbit, hence the automaton cannot be thin.

For all the other combinations we just exhibit as examples the automata from Fig. 3:

- $\mathcal{A}_6$ is thin and meager;
- $\mathcal{A}_{10}$ is thin and normal;

- $\mathcal{A}_8$ is thin and obese (type II);
- $\mathcal{A}_4$ is thick and normal;
- $\mathcal{A}_1$ is thick and obese (type I).
- $\mathcal{A}_2$ is thick and obese (type II). ◀

6 Conclusion

We defined three classes of bandwidth and two structural criteria to classify automata into these three classes. We believe these classes will serve in the theory of timed automata since they correspond to very different kinds of behavior. We believe that techniques specific to the class of automaton would be useful for the random generation of its runs, and improve the performance and scalability of tools such as Wordgen [6]. Our results have been formulated for timed words (time event sequences), and we think that porting them to timed signals would bring interesting insights. We believe that our results can be rephrased in terms of Kolmogorov complexity of ε-approximations of timed words.

In the future, we would like to be able to do more than classify automata: we would also like to compute the constant α that multiplies the main term in the bandwidth formulas. This step will be essential for applications to timed information coding, in the spirit of [2]. We hope that many tools introduced in this article will serve to reach this ambitious aim.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994. doi:10.1016/0304-3975(94)90010-8.
2. Eugene Asarin, Nicolas Basset, Marie-Pierre Béal, Aldric Degorre, and Dominique Perrin. Toward a timed theory of channel coding. In *Proc. FORMATS*, volume 7595 of *LNCS*, pages 27–42, 2012. doi:10.1007/978-3-642-33365-1_4.
3. Eugene Asarin, Nicolas Basset, and Aldric Degorre. Entropy of regular timed languages. *Information and Computation*, 241:142–176, 2015. doi:10.1016/j.ic.2015.03.003.
4. Eugene Asarin, Nicolas Basset, and Aldric Degorre. Distance on timed words and applications. In *Proc. FORMATS*, volume 11022 of *LNCS*, pages 199–214, 2018. doi:10.1007/978-3-030-00151-3_12.
5. Eugene Asarin, Aldric Degorre, Cătălin Dima, and Bernardo Jacobo Inclán. Bandwidth of timed automata: 3 classes, 2023. arXiv:2310.01941.
6. Benoît Barbot, Nicolas Basset, and Alexandre Donzé. Wordgen: a timed word generation tool. In *Proc. HSCC*, pages 16:1–16:7. ACM, 2023. doi:10.1145/3575870.3587116.
7. Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and Automata*. Cambridge University Press, 2009.
8. Mikołaj Bojańczyk. Factorization forests. In *Proc. DLT*, volume 5583 of *LNCS*, pages 1–17, 2009. doi:10.1007/978-3-642-02737-6_1.
9. Martin R. Bridson and Robert H. Gilman. Context-free languages of sub-exponential growth. *Journal of Computer and System Sciences*, 64(2):308–310, 2002. doi:10.1006/jcss.2001.1804.
10. Noam Chomsky and George A. Miller. Finite state languages. *Information and Control*, 1(2):91–112, 1958. doi:10.1016/S0019-9958(58)90082-2.
11. Paweł Gawrychowski, Dalia Krieger, Narad Rampersad, and Jeffrey Shallit. Finding the growth rate of a regular or context-free language in polynomial time. In *Proc. DLT*, volume 5257 of *LNCS*, pages 339–358, 2008. doi:10.1007/978-3-540-85780-8_27.
12. Rostislav Grigorchuk and Antonio Machí. An example of an indexed language of intermediate growth. *Theoretical Computer Science*, 215(1):325–327, 1999. doi:10.1016/S0304-3975(98)00161-3.

13 Vineet Gupta, Thomas A. Henzinger, and Radha Jagadeesan. Robust timed automata. In *Hybrid and Real-Time Systems*, volume 1201 of *LNCS*, pages 331–345. Springer, 1997. doi:10.1007/BFb0014736.

14 K.A.S. Immink. EFMPlus: The coding format of the multimedia compact disc. *IEEE Transactions on Consumer Electronics*, 41(3):491–497, 1995. doi:10.1109/30.468040.

15 K.A.S. Immink. *Codes for Mass Data Storage Systems*. Shannon Foundation Publ., 2004.

16 Bernardo Jacobo Inclán, Aldric Degorre, and Eugene Asarin. Bounded delay timed channel coding. In *Proc. FORMATS*, volume 13465 of *LNCS*, pages 65–79, 2022. doi:10.1007/978-3-031-15839-1_4.

17 Andrei Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968. doi:10.1080/00207166808803030.

18 Andrei Kolmogorov and Vladimir Tikhomirov. ε-entropy and ε-capacity of sets in function spaces. *Uspekhi Matematicheskikh Nauk*, 14(2):3–86, 1959. doi:10.1007/978-94-017-2973-4_7.

19 Douglas Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.

20 Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

21 Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000. doi:10.1023/A:1008387132377.

22 John Rhodes and Pedro V. Silva. *Boolean Representations of Simplicial Complexes and Matroids*. Springer International Publishing, 2015. doi:10.1007/978-3-319-15114-4.

23 Ocan Sankur. *Robustness in timed automata: analysis, synthesis, implementation*. PhD thesis, École normale supérieure de Cachan, Paris, France, 2013. URL: https://tel.archives-ouvertes.fr/tel-00910333.

24 Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

25 Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990. doi:10.1016/0304-3975(90)90047-L.

26 Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991. doi:10.1016/0304-3975(91)90381-B.

Monotone Classes Beyond VNP

Prerona Chatterjee
Blavatnik School of Computer Science, Tel Aviv University, Israel

Kshitij Gajjar
Indian Institute of Technology Jodhpur, Rajasthan, India

Anamay Tengse
Department of Computer Science, University of Haifa, Israel

Abstract

In this work, we study the natural monotone analogues of various equivalent definitions of VPSPACE: a well studied class (Poizat 2008, Koiran & Perifel 2009, Malod 2011, Mahajan & Rao 2013) that is believed to be larger than VNP. We observe that these monotone analogues are not equivalent unlike their non-monotone counterparts, and propose *monotone* VPSPACE (mVPSPACE) to be defined as the monotone analogue of Poizat's definition. With this definition, mVPSPACE turns out to be exponentially stronger than mVNP and also satisfies several desirable closure properties that the other analogues may not.

Our initial goal was to understand the monotone complexity of *transparent polynomials*, a concept that was recently introduced by Hrubeš & Yehudayoff (2021). In that context, we show that transparent polynomials of large sparsity are hard for the monotone analogues of all the known definitions of VPSPACE, except for the one due to Poizat.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic Complexity, Monotone Computation, VPSPACE, Transparent Polynomials

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.11

Related Version *Full Version on Arxiv*: `https://arxiv.org/abs/2202.13103`

Funding *Prerona Chatterjee*: Azrieli International Postdoctoral Fellowship, the Israel Science Foundation (grant number 514/20) and the Len Blavatnik and the Blavatnik Family foundation. Parts of this work was done while I was a PhD student at TIFR Mumbai, where I was partially supported by a Google PhD Fellowship and also while I was a postdoctoral researcher at the Czech Academy of Sciences, where I was supported by Grant GX19-27871X of the Czech Science Foundation.
Kshitij Gajjar: Part of this work was done as a postdoc in NUS, where I was funded by NUS ODPRT Grant WBS No. R-252-000-A94-133.
Anamay Tengse: Grant 716/20 of the Israel Science Foundation.

Acknowledgements We would like to thank the anonymous reviewers for the helpful comments. We are also grateful to Meena Mahajan for suggestions that greatly improved the presentation of the paper.

1 Introduction

The aim of algebraic complexity is to classify polynomials in terms of how hard it is to compute them, and the most standard model for computing polynomials is that of an *algebraic circuit*. An algebraic circuit is a rooted, directed acyclic graph where the leaves are labelled with variables or field constants and internal nodes are labelled with addition (+) or multiplication (×). Every node therefore naturally computes a polynomial and the polynomial computed by the root is said to be the polynomial computed by the circuit. A formal definition can be

© Prerona Chatterjee, Kshitij Gajjar, and Anamay Tengse;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 11; pp. 11:1–11:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

found in Section 2. The central question in the area is to show super-polynomial lower bounds against algebraic circuits for *explicit* polynomials, or equivalently, to show that $\mathsf{VP} \neq \mathsf{VNP}$: the algebraic analogue of the famed P vs. NP question.

However, proving strong lower bounds against circuits has turned out to be a difficult problem. Much of the research therefore naturally focusses on various restricted algebraic models which compute correspondingly structured polynomials. One such syntactic restriction is that of *monotonicity*, where the models are not allowed to use any negative constants. Therefore, trivially, monotone circuits always compute polynomials with only non-negative coefficients. Such polynomials are called *monotone polynomials*. We denote the class of all polynomials that are efficiently computable by monotone algebraic circuits by mVP. Also note that any monomial computed during intermediate computation in a monotone circuit can never get cancelled out, making it a fairly weak model. As a result, several strong lower bounds are known against monotone circuits.

Lower bounds in the monotone setting

There has been a long line of classical works that prove lower bounds against monotone algebraic circuits [21, 22, 23, 12, 13, 7]. The most well-known among these, is the result of Jerrum & Snir [12], where they showed exponential lower bounds against monotone circuits for many polynomial families including the Permanent (Perm_n). In particular, they showed that every monotone algebraic circuit computing the n^2-variate Perm_n must have size at least $2^{\Omega(n)}$. A few of the more recent works on monotone lower bounds include [20, 6, 1].

Additionally, many separations that are believed to be true in the general setting have actually been proved to be true in the monotone setting [22, 9, 26, 24]. Most remarkably, Yehudayoff [26] showed an exponential separation between the computational powers of the monotone analogues of VP and VNP. We denote these classes by mVP (Definition 2.3) and mVNP (Definition 2.4) respectively.

Another line of work in this setting tries to understand the power of non-monotone computational models while computing monotone polynomials. Valiant [25], in his seminal paper, showed that there is a family of monotone polynomials which can be computed by polynomial sized non-monotone algebraic circuits such that any monotone algebraic circuit computing them must have exponential size. More recent works [10, 4, 3, 5] have shown even stronger separations between the relative powers of monotone and non-monotone models while computing monotone polynomials.

Newton polytopes, transparency and monotone complexity

Returning briefly to the general setting, an interesting conjecture relating the algebraic complexity of a bivariate polynomial to its geometric property is the "Tau-conjecture" (also written as τ-conjecture). The Newton polytope of an n-variate polynomial f, denoted by $\mathsf{Newt}(f)$, is the convex hull in $\mathbb{R}^n$ of the *exponent vectors* of the monomials in the support of f. Recently, Hrubeš & Yehudayoff [11] proposed studying the *Shadows of Newton polytopes* (projections to two-dimensional planes) as an approach to refute the τ-conjecture for Newton polygons made by Koiran, Portier, Tavenas & Thomassé [16].

Informally, the τ-conjecture for Newton polygons [16] states that if f is a bivariate polynomial that can be written as an s-sum of r-products of p-sparse polynomials, then its Newton polygon has at most $\mathrm{poly}(s, r, p)$ vertices. A formal definition of Newton polytopes and the τ-conjecture for Newton polygons can be found in Appendix A.

This conjecture is fairly strong, and it implies, among other things, that $\mathsf{VP} \neq \mathsf{VNP}$. However, observe that the Newton polygon retains no information about the coefficients of the polynomial. Since the algebraic complexity of polynomials is believed to be heavily

dependent on coefficients (for example the determinant (Det_n) is efficiently computable by algebraic circuits and this is expected to not be the case for Perm_n, even though they have the same set of monomials), the τ-conjecture for Newton polygons is believed to be false.

The approach suggested by Hrubeš & Yehudayoff [11] used shadows of Newton polytopes as a means to move from the multivariate setting to the bivariate setting, and use polynomials like determinant (Det_n) to refute the conjecture. The difficulty in this strategy however, is to find a polynomial in VP that exhibits high *shadow complexity* (maximum number of vertices in its projection), since even when a candidate polynomial is fixed, say Det_n, it is not easy to design a suitable bivariate projection.

As a means to tackle this issue, Hrubeš & Yehudayoff introduced the notion of *transparent polynomials* – polynomials that can be projected to bivariates in such a way that all of their monomials become vertices of the resulting Newton polygon. Further, they also gave examples of polynomials with exponentially large sets of monomials that are provably transparent. Therefore, a proof of any one of these polynomials being in VP would directly refute the τ-conjecture for Newton polytopes.

Even though Hrubeš and Yehudayoff [11] were not able to actually use this approach to refute the conjecture, they used the notions of shadows & transparency to come up with yet another method for proving lower bounds against monotone algebraic circuits. They showed that the monotone circuit complexity of a polynomial is lower bounded by its shadow complexity when the polynomial is transparent.

▶ **Theorem 1.1** ([11, Theorem 2]). *If f is transparent then every monotone circuit computing f has size at least $\Omega(|\mathrm{supp}(f)|)$.*

As a corollary, they present an n-variate polynomial such that any monotone algebraic circuit computing it must have size $\Omega(2^{n/3})$.

1.1 Our Contribution

Here we state our contributions informally; the formal statements can be found in Section 3. Throughout this work we assume that the underlying field is either the field of real numbers or the field of rational numbers. The goal of this work is two-fold.

The first goal is to understand how restrictive the notion of transparency is. Our search begins with an observation by Yehudayoff [26], that any lower bound against mVP depending solely on the support of the hard polynomial, automatically "lifts" to mVNP with the same parameters[1]. Since transparency is a property solely of the Newton polytope, and hence of the support of the polynomial, the above observation shows that any transparent polynomial that is non-sparse (has super-polynomially large support) is hard to compute even for mVNP. However, we suspect that transparency is an even stronger property. Therefore, a natural question for us is whether there are even larger classes of monotone polynomials that do not contain non-sparse, transparent polynomials.

This brings us to the second goal of this work – studying monotone models of computation that can possibly compute polynomials outside even mVNP. Classes larger than VNP had not been defined in the monotone world prior to this work. We therefore turn to the literature in the non-monotone setting. Here, $\mathsf{VPSPACE}$ is a well studied class [19, 14, 18, 17]

[1] [26]: "If a monotone circuit-size lower bound for $q(\mathbf{x})$ holds also for all polynomials that are equivalent to $q(\mathbf{x})$ then the same lower bound also holds for every mVNP circuit computing $q(\mathbf{x})$." Here mVNP circuit denotes $\sum_{\mathbf{z}\in\{0,1\}^m} \mathcal{C}(\mathbf{x}, z_1, \ldots, z_m)$ where $m = \mathrm{poly}(n)$ and $\mathcal{C}(\mathbf{x}, \mathbf{z})$ is a monotone algebraic circuit.

that is believed to be strictly larger than VNP. Interestingly there are multiple definitions of VPSPACE, resulting from varied motivations, all of which are known to be essentially equivalent [18, 17]. We study the natural monotone analogues of these definitions and show that unlike the non-monotone setting, the powers of the different resulting models vary greatly. This allows us to then analyse if the technique of Hrubeš & Yehudayoff also works against monotone classes that are possibly larger than mVNP.

The following figure succinctly describes some of our main results.

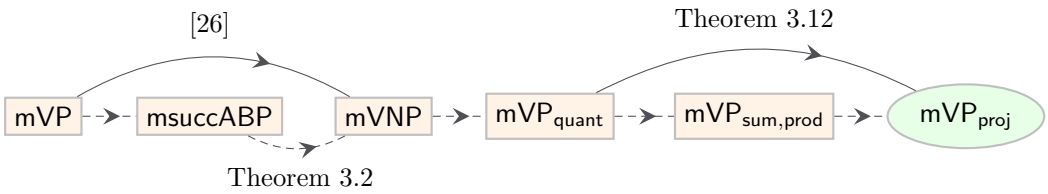


Figure 1 Nodes represent classes of polynomial families; $A \dashrightarrow B \equiv A \subseteq B$ and $A \longrightarrow B \equiv A \subsetneq B$. Transparent polynomials are hard for all models corresponding to orange, rectangular nodes.

In Figure 1, the node labels refer to the following classes of polynomial families that have degree-poly(n) and poly(n)-complexity under the corresponding models.

- msuccABP - monotone succinct ABPs (Definition 3.1),
- $\mathsf{mVP}_{\mathsf{quant}}$ - quantified monotone circuits (Definition 3.4),
- $\mathsf{mVP}_{\mathtt{sum,prod}}$ - monotone circuits with summation and production gates (Definition 3.8),
- $\mathsf{mVP}_{\mathsf{proj}}$ - monotone circuits with projection gates (Definition 3.11).

The orange, rectangular nodes denote the classes in which sparsity of transparent polynomials in it is bounded by a constant factor of the size of the smallest $\mathcal{M}$ computing it, if $\mathcal{M}$ is the computational model corresponding to the class (Theorem 3.10).

An interesting point to note here is that there is an exponential separation between $\mathsf{mVP}_{\mathsf{quant}}$ and $\mathsf{mVP}_{\mathsf{proj}}$, which means that at least one of the inclusions: $\mathsf{mVP}_{\mathsf{quant}}$ to $\mathsf{mVP}_{\mathsf{sum,prod}}$, and $\mathsf{mVP}_{\mathsf{sum,prod}}$ to $\mathsf{mVP}_{\mathsf{proj}}$ is strict with an exponential separation.

Additionally, we show the following two statements about $\mathsf{mVP}_{\mathsf{quant}}$.

- $\mathsf{mVP}_{\mathsf{quant}} = \mathsf{mVNP}$ if and only if homogeneous components of polynomials in $\mathsf{mVP}_{\mathsf{quant}}$ are contained in $\mathsf{mVP}_{\mathsf{quant}}$ (Corollary 3.6). In particular, we show that homogeneous polynomials in $\mathsf{mVP}_{\mathsf{quant}}$ are also in mVNP (Theorem 3.5).
- $\mathsf{mVP}_{\mathsf{quant}} = \mathsf{mVP}_{\mathsf{sum,prod}}$ if and only if quantified monotone circuits are closed under compositions (Observation 3.9).

Finally, we also show that the homogeneous components of polynomials in $\mathsf{mVP}_{\mathsf{proj}}$ are in $\mathsf{mVP}_{\mathsf{proj}}$ (Theorem 3.13). This property, along with the fact that $\mathsf{Perm}_n \in \mathsf{mVP}_{\mathsf{proj}}$ (Theorem 6.1), is the reason we propose "monotone VPSPACE" (mVPSPACE) to be defined as the class of polynomial families that are efficiently computable by monotone circuits with projection gates (without any restriction on degree).

1.2 Organization of the paper

We begin in Section 2 with formal definitions for all the models of computation that we will be using. Next, we define the monotone analogues of the various definitions of VPSPACE, and outline our results about them in Section 3. The proofs of our results are discussed in Section 4, Section 5 and Section 6. We conclude with Section 7, where we discuss some important open threads from our work.

2 Preliminaries

We shall use the following notation for the rest of the paper.

- We use the standard shorthand $[n] = \{1, 2, \ldots, n\}$.
- We use boldface letters like $\mathbf{x}, \mathbf{z}, \mathbf{e}$ to denote tuples/sets of variables or constants, individual members are expressed using indexed version of the usual symbols: $\mathbf{e} = (e_1, e_2, \ldots, e_n)$, $\mathbf{x} = \{x_1, \ldots, x_n\}$. We also use $|\mathbf{y}|$ to denote the size/length of a vector $\mathbf{y}$.
  For vectors $\mathbf{x}$ and $\mathbf{e}$ of the same length n, we use the shorthand $\mathbf{x}^{\mathbf{e}}$ to denote the monomial $x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$.
- For a polynomial $f(\mathbf{x})$, we denote by $\deg(f)$ the degree of f in $\mathbf{x}$.
- For a polynomial $f(\mathbf{x})$ and a monomial $m = \mathbf{x}^{\mathbf{e}}$, we refer to the coefficient of m in f by $\mathrm{coeff}_f(m)$. The support $\mathrm{supp}(f)$ of a polynomial f is given by $\{m : \mathrm{coeff}_f(m) \neq 0\}$, and the *sparsity* of a polynomial is the size of its support, $|\mathrm{supp}(f)|$.
- For any polynomial $f(\mathbf{x})$ and any $k \leq \deg(f)$, we denote by $\mathrm{hom}_k(f)$ the k-th homogeneous degree component of f in terms of $\mathbf{x}$. That is, if $f(\mathbf{x}) = f_0(\mathbf{x}) + \ldots + f_{\deg(f)}(\mathbf{x})$ where $f_k(\mathbf{x})$ is a homogeneous polynomial of degree k in $\mathbf{x}$, then $\mathrm{hom}_k(f) = f_k$.
- The permanent of an $n \times n$ symbolic matrix shall be denoted by Perm_n and is defined as $\mathsf{Perm}_n = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$, where S_n is the set of all permutations of $[n]$.
- We use $\{f_n\}$ to denote families of polynomials indexed by $\mathbb{N}$. All complexity classes are defined in terms of asymptotic properties of "polynomials" and are technically sets of such polynomial families. Sometimes however, this technicality is ignored for the sake of brevity, especially when the analogous statement about polynomial families is obvious.

▶ **Definition 2.1** (Algebraic circuits). *An algebraic circuit is a directed acyclic graph with leaves (nodes with in-degree zero) labelled by formal variables and constants from the field, and other nodes labelled by addition* $(+)$ *and multiplication* $(\times)$ *have in-degree 2.*

The leaves compute their labels, and every other node computes the operation it is labelled by, on the polynomials along its incoming edges. A node of out-degree zero is called the output of the circuit, and the circuit is said to compute the polynomial computed by the output gate.

In case there are multiple output gates, the circuit is said to be multi-output*, and computes a set of polynomials.*

The size *of a circuit,* $\mathcal{C}$*, denoted by* $\mathrm{size}(\mathcal{C})$*, is the number of nodes in the graph.*

An algebraic circuit over $\mathbb{Q}$ *or* $\mathbb{R}$ *is said to be* monotone*, if all the constants appearing in it are non-negative.*

▶ **Definition 2.2** (Algebraic Branching Programs (ABPs)). *An* algebraic branching program *is specified by a layered graph where each edge is labelled by an affine linear form and the first and the last layer have one vertex each, called the "source" and the "sink" vertex respectively. The polynomial computed by an ABP is equal to the sum of the weights of all paths from the start vertex to the end vertex in the ABP, where the weight of a path is equal to the product of the labels of all the edges on it.*

The width of a layer in an ABP is the number of vertices in it and the width of an ABP is the width of the layer that has the maximum number of vertices in it.

The size of an ABP is the number of vertices in it.

Basic monotone classes

▶ **Definition 2.3** (Monotone VP (mVP)). *A family* $\{f_n\}$ *of monotone polynomials is said to be in* mVP*, if there exists a constant* $c \in \mathbb{N}$ *such that for all large* n*,* f_n *depends on at most* n^c *variables, has degree at most* n^c*, and is computable by a monotone algebraic circuit of size at most* n^c*.*

▶ **Definition 2.4** (Monotone VNP (mVNP)). *A family $\{f_n\}$ of monotone polynomials is said to be in* mVNP*, if there exists a constant $c \in \mathbb{N}$, and an m-variate family $\{g_m\} \in$ mVP with $m, \text{size}(g_m) \leq n^c$, such that for all large enough n, f_n satisfies the following.*

$$f_n(\mathbf{x}) = \sum_{\mathbf{a} \in \{0,1\}^{|\mathbf{y}|}} g_m(\mathbf{x}, \mathbf{y} = \mathbf{a})$$

An expression of the above form is alternatively called *an exponential sum* computing f_n.

Various definitions of VPSPACE

Koiran & Perifel [14, 15] were the first to define VPSPACE as the class of polynomials (of degree that is potentially exponential in the number of underlying variables) whose coefficients can be computed in PSPACE/ poly, and $\mathsf{VPSPACE}_\mathrm{b}$ to be the polynomials in VPSPACE that have degree bounded by a polynomial in the number of underlying variables. They showed that if $\mathsf{VP} \neq \mathsf{VPSPACE}_\mathrm{b}$ then either $\mathsf{VP} \neq \mathsf{VNP}$ or $\mathsf{P}/\,\text{poly} \neq \mathsf{PSPACE}/\,\text{poly}$.

Later, Poizat [19] gave an alternate definition that does not rely on any boolean machinery, but instead uses a new type of gate called a *projection gate*.

▶ **Definition 2.5** (Projection gates [19]). *A* projection *gate is a* unary *gate that is labelled by a variable z and a constant $b \in \{0, 1\}$, denoted by $\mathtt{fix}_{(z=b)}$. It returns the partial evaluation of its input polynomial, at $z = b$, that is, $\mathtt{fix}_{(z=b)}(f(z, \mathbf{x})) = f(b, \mathbf{x})$.*

Poizat defined algebraic circuits with projection gates and then defined VPSPACE to be the class of polynomial families that are efficiently computable by this model. Poizat showed[2] that this definition is equivalent to that of Koiran & Perifel.

▶ **Definition 2.6** (Algebraic circuits with projection gates [19]). *An* algebraic circuit with projection gates *is an algebraic circuit (Definition 2.1) in which the internal nodes can also be projection gates (Definition 2.5), in addition to $+$ or $\times$.*

The size of an algebraic circuit with projection gates is the number of nodes in the underlying graph.

Adding to Poizat's work, Malod [18] characterized VPSPACE using exponentially large *algebraic branching programs (ABPs)* that are *succinct*. Malod's work defines the *complexity* of an ABP as the size of the smallest algebraic circuit that encodes its graph – outputs the corresponding edge label when given the two endpoints as input. An n-variate ABP is then said to be *succinct*, if its complexity is $\text{poly}(n)$.

▶ **Definition 2.7** (Succinct ABPs [18]). *A* succinct ABP *over the n variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ is a triple $(B, \mathbf{s}, \mathbf{t})$ with $|\mathbf{s}| = |\mathbf{t}| = r$, where*

- *$\mathbf{s}$ is the label of the source vertex, and $\mathbf{t}$ is the label of the sink(target) vertex.*
- *$B(\mathbf{u}, \mathbf{v}, \mathbf{x})$ is an algebraic circuit that describes a directed acyclic graph G_B on the vertex set $\{0, 1\}^r$ in the following way. For any two vertices $\mathbf{a}, \mathbf{b} \in \{0, 1\}^r$, the output $B(\mathbf{u} = \mathbf{a}, \mathbf{v} = \mathbf{b}, \mathbf{x})$ is the label of the edge from $\mathbf{a}$ to $\mathbf{b}$ in the ABP.*

The polynomial computed by the ABP is the sum of polynomials computed along all $\mathbf{s}$ to $\mathbf{t}$ paths in G_B; where each path computes the product of the labels of the constituent edges.

The size of the circuit B is said to be the complexity *of the succinct ABP. The number of vertices 2^r is the* size *of the succinct ABP, and the length of the longest $\mathbf{s}$ to $\mathbf{t}$ path is called the* length *of the succinct ABP.*

[2] The work of Poizat is written in French, Malod [18] provides an alternate exposition of some of the main results in English.

In the same work [18], Malod alternatively characterized VPSPACE using an interesting algebraic model that resembles *(totally) quantified boolean formulas* that are known to characterize PSPACE. This model, which we refer to as "quantified algebraic circuits", is defined using special types of projection gates called *summation* and *production* gates.

▶ **Definition 2.8** (Summation and Production gates [18]). Summation *and* production *gates are unary gates that are labelled by a variable z, and are denoted by* $\mathtt{sum}_z$ *and* $\mathtt{prod}_z$ *respectively. A summation gate returns the sum of the* $(z = 0)$ *and* $(z = 1)$ *evaluations of its input, and a production gate returns the product of those evaluations. That is,* $\mathtt{sum}_z(f(z, \mathbf{x})) = f(0, \mathbf{x}) + f(1, \mathbf{x})$*, and* $\mathtt{prod}_z(f(z, \mathbf{x})) = f(0, \mathbf{x}) \cdot f(1, \mathbf{x})$*.*

We sometimes use $\mathtt{sum}_{\{z_1,\ldots,z_k\}}$ *to refer to the nested expression* $\mathtt{sum}_{z_1} \cdots \mathtt{sum}_{z_k}$ *(similarly for* $\mathtt{prod}$*); it can be checked that the order does not matter here.*

A quantified algebraic circuit has the form $\mathtt{Q}^1_{z_1}\mathtt{Q}^2_{z_2}\cdots\mathtt{Q}^m_{z_m}\mathcal{C}(\mathbf{x}, \mathbf{z})$, where each $\mathtt{Q}^i$ is a summation or a production, and $\mathcal{C}(\mathbf{x}, \mathbf{z})$ is a usual algebraic circuit.

▶ **Definition 2.9** (Quantified Algebraic Circuits [18]). *A quantified algebraic circuit is an algebraic circuit that has the form,*

$$\mathtt{Q}^{(1)}_{z_1}\mathtt{Q}^{(2)}_{z_2}\cdots\mathtt{Q}^{(m)}_{z_m}\mathcal{C}(\mathbf{x}, \mathbf{z}),$$

where $|\mathbf{z}| = m$*,* $\mathtt{Q}^{(i)} \in \{\mathtt{sum}, \mathtt{prod}\}$ *for each* $i \in [m]$*, and* $\mathcal{C}$ *is an algebraic circuit. The size of such a quantified algebraic circuit is* $m + \mathrm{size}(\mathcal{C})$*.*

Finally, Mahajan & Rao [17] defined algebraic analogues of small space computation (e.g. L, NL) using the notion of *width* of an algebraic circuit. They use their definitions to import some relationships known in the boolean world to the algebraic world (e.g, they show VL $\subseteq$ VP). They further show that their definition of uniform polynomially-bounded-space computation coincides with that of uniform-VPSPACE as defined by Koiran & Perifel [14].

We now narrow our focus to the definitions due to Poizat [19] and Malod [18]. We choose these definitions because they are algebraic in nature, and have fairly natural monotone analogues. We elaborate a bit more about this decision in Appendix B.

▶ Remark. It should be noted that all the above-mentioned definitions of VPSPACE allow for the polynomial families to have large degree – as high as $\exp(\mathrm{poly}(n))$. The main focus of our work, however, is to compare the monotone analogues of these models with mVP and mVNP. Since the latter classes only contain low-degree polynomials, we will only work with polynomials of degree $\mathrm{poly}(n)$, or $\mathsf{VPSPACE}_{\mathrm{b}}$ as defined in [14], unless mentioned otherwise.

3 Monotone analogues of VPSPACE, and our contributions

We now define monotone analogues for the various definitions of VPSPACE outlined in the previous section, and compare the powers of the resulting monotone models/classes.

3.1 Monotone succinct ABPs

We first consider the natural monotone analogue of the definition due to Malod [18] which uses succinct algebraic branching programs (Definition 2.7).

Malod showed that every family $\{f_n\}$ in VPSPACE can be computed by $2^{\mathrm{poly}(n)}$ sized ABPs that have *complexity* $\mathrm{poly}(n)$. Recall that the complexity of a succinct ABP is the size of the smallest algebraic circuit that encodes its graph.

We therefore define monotone succinct ABPs as ABPs that can be succinctly described by *monotone* algebraic circuits of size poly(n). However, this restriction forces that if the monomial $\mathbf{x}^{\mathbf{e}}$ appears in any edge-label $(\mathbf{a}, \mathbf{b})$, then it also appears in the label of $(\bar{1}, \bar{1})$. Therefore, self-loops are inevitably present in succinct ABPs in the monotone setting. To handle this, we additionally allow the *length* of the ABP, say ℓ, to be predefined[3] so that now the polynomial computed by the ABP can be defined to be the sum of polynomials computed by all $\mathbf{s} - \mathbf{t}$ paths of length at most ℓ.

▶ **Definition 3.1** (Monotone Succinct ABPs). *A* monotone succinct ABP *over the n variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ is a four tuple $(B, \mathbf{s}, \mathbf{t}, \ell)$ with $|\mathbf{s}| = |\mathbf{t}| = r$, where*

- *ℓ is the* length *of the ABP.*
- *$\mathbf{s}$ is the label of the source vertex, and $\mathbf{t}$ is the label of the sink (target) vertex.*
- *$B(\mathbf{u}, \mathbf{v}, \mathbf{x})$ is a* monotone *algebraic circuit that describes a directed graph G_B on the vertex set $\{0,1\}^r$ in the following way. For any two vertices $\mathbf{a}, \mathbf{b} \in \{0,1\}^r$, the output $B(\mathbf{u} = \mathbf{a}, \mathbf{v} = \mathbf{b}, \mathbf{x})$ is the label of the edge from $\mathbf{a}$ to $\mathbf{b}$ in the ABP.*

The polynomial computed by the ABP is the sum of polynomials computed along all $\mathbf{s}$ to $\mathbf{t}$ paths in G_B of length at most ℓ; where each path computes the product of the labels of the constituent edges.

The size of the circuit B is said to be the complexity *of the monotone succinct ABP. The number of vertices 2^r is the* size *of the succinct ABP.*

Note that since B is a monotone algebraic circuit, all the edge-labels in the ABP are monotone polynomials over $\mathbf{x}$. It is also not hard to see that any polynomial $f \in \mathsf{mVP}$ is computable by this model. If $\mathcal{C}$ is the monotone circuit computing f, then the monotone succinct ABP computing f is $(\mathcal{C}', 0, 1, 1)$ where $\mathcal{C}'(u, v, \mathbf{x}) = v \cdot \mathcal{C}(\mathbf{x})$.

We show that the computational power of monotone succinct ABPs when computing polynomials of *bounded degree* does not even go beyond mVNP.

▶ **Theorem 3.2.** *If a polynomial family $\{f_n\}$ of degree* poly(n) *is computable by monotone succinct ABPs of complexity* poly(n)*, then $\{f_n\} \in \mathsf{mVNP}$.*

In contrast, Malod [18] showed that every family in $\mathsf{VPSPACE}$ admits succinct ABPs of polynomial complexity, and we expect $\mathsf{VPSPACE}_b$ to be a much bigger class than VNP. The proof of Theorem 3.2 is quite straightforward and relies on the following claim.

▷ Claim 3.3. For any f_n, let $\mathcal{A} = (B, \mathbf{s}, \mathbf{t}, \ell)$ be the monotone succinct ABP computing it, with $|\mathbf{s}| = |\mathbf{t}| = r$. If $\ell > 1$, then $\ell \leq \deg(f_n) + 2$.

This bound allows us to write the sum of all s to t paths in the ABP as an exponential sum of an mVP expression, finishing the proof. A complete proof can be found in Subsection C.4.

It is not clear to us if the converse of Theorem 3.2 is true. Any obvious attack seems to fail due to the restriction that the circuit encoding the ABP needs to be monotone.

3.2 Quantified monotone circuits

As mentioned earlier, Malod [18] had also characterized the class $\mathsf{VPSPACE}$ using the notion of quantified algebraic circuits (Definition 2.9). We now consider its natural monotone analogue, which we call quantified monotone circuits.

[3] It is not hard to see that the analogous definition in the non-monotone setting is equivalent to Malod's definition (Definition 2.7). This is essentially because of the connection to Iterated Matrix Multiplication.

▶ **Definition 3.4** (Quantified Monotone Algebraic Circuits). *A quantified monotone algebraic circuit has the form*

$$\mathtt{Q}^{(1)}_{z_1}\mathtt{Q}^{(2)}_{z_2}\cdots\mathtt{Q}^{(m)}_{z_m}\mathcal{C}(\mathbf{x},\mathbf{z})$$

where $|\mathbf{z}| = m$, $\mathtt{Q}^{(i)} \in \{\mathtt{sum}, \mathtt{prod}\}$ *for each* $i \in [m]$, *and* $\mathcal{C}$ *is a monotone algebraic circuit. The size of the quantified monotone algebraic circuit above is* $m + \mathrm{size}(\mathcal{C})$.

We denote by $\mathsf{mVP}_{\mathsf{quant}}$ *the class of all* n*-variate polynomial families of degree* $\mathrm{poly}(n)$ *that are computable by quantified monotone algebraic circuits of size* $\mathrm{poly}(n)$.

Clearly $\mathsf{mVNP} \subseteq \mathsf{mVP}_{\mathsf{quant}}$. It is therefore interesting to check if the inclusion is tight. We show that $\mathsf{mVNP} \neq \mathsf{mVP}_{\mathsf{quant}}$ if and only if there is a family $\{f_n\} \in \mathsf{mVP}_{\mathsf{quant}}$ such that the k-th homogeneous component of f_n is not in $\mathsf{mVP}_{\mathsf{quant}}$ for some n and $k \leq \deg(f)$.

In particular we show the following statement.

▶ **Theorem 3.5.** *Let* f *be computable by a quantified monotone circuit of size* s. *If* f *is homogeneous, then it is expressible as an exponential sum of size at most* $O(s \cdot \deg(f))$.

Since mVNP is closed under addition, we get the following as a corollary.

▶ **Corollary 3.6.** *The class* $\mathsf{mVP}_{\mathsf{quant}}$ *is closed under taking homogeneous components, if and only if,* $\mathsf{mVP}_{\mathsf{quant}} = \mathsf{mVNP}$. *That is,*

$$(\forall f \in \mathsf{mVP}_{\mathsf{quant}}, \forall k \leq \deg(f), \mathrm{hom}_k(f) \in \mathsf{mVP}_{\mathsf{quant}}) \iff \mathsf{mVNP} = \mathsf{mVP}_{\mathsf{quant}}$$

A proof of Theorem 3.5 and Corollary 3.6 can be found in Section 4.

Even though we believe $\mathsf{mVNP} \subsetneq \mathsf{mVP}_{\mathsf{quant}}$, we feel this might be tricky to prove. The following theorem sheds some light on why that may be the case.

▶ **Theorem 3.7.** *Suppose* $f(\mathbf{x})$ *is an* n*-variate, degree-*d *polynomial computed by a quantified monotone circuit of size* s, *which uses* ℓ *summation gates. Then for a set of variables* $\mathbf{w}$ *of size at most* $d \cdot \ell$, *there is a monotone circuit* $h(\mathbf{x}, \mathbf{w})$ *of size at most* $d \cdot s$, *and a monotone polynomial* $A(\mathbf{w})$ *such that,*

$$f(\mathbf{x}) = \sum_{\mathbf{b} \in \{0,1\}^{|\mathbf{w}|}} A(\mathbf{w} = \mathbf{b}) \cdot h(\mathbf{x}, \mathbf{w} = \mathbf{b}), \tag{1}$$

where $A(\mathbf{w})$ *potentially has circuit size and degree that is exponential in* n *and* ℓ.

Although the obvious size and degree bounds on $A(\mathbf{w})$ above are exponential, it has a somewhat succinct quantified expression that can be inferred from the proof (see Subsection C.5). We now discuss how Theorem 3.7 helps us understand a possible difficulty in separating $\mathsf{mVP}_{\mathsf{quant}}$ from mVNP.

1. If the polynomial $A(\mathbf{w})$ from Theorem 3.7 were to have degree and size that is polynomial in n, then $\mathsf{mVP}_{\mathsf{quant}}$ would collapse to mVNP. Further, since A is free of $\mathbf{x}$, its exponential degree and size can be leveraged only for designing coefficients of f. Moreover, the monotone nature of A and h ensures that $A(\mathbf{1})$ is the largest value, and contributes *equally* to all monomials in the support of f, since $\mathrm{supp}(f) = \mathrm{supp}(h(\mathbf{x}, \mathbf{w} = \mathbf{1}))$.
2. Another consequence that is quite interesting is the following. Suppose there is a different monotone polynomial $B(\mathbf{w})$ of small degree and size that agrees with $A(\mathbf{w})$ on all $\{0,1\}$-inputs, then $f(\mathbf{x}) = \sum_{\mathbf{b}} B(\mathbf{b})h(\mathbf{x}, \mathbf{b})$. That is, we can replace A by B in our expression and then f clearly has an efficient "mVNP-expression".
   Thus, any separation between mVNP and quantified monotone VP will provide a polynomial $A(\mathbf{w})$ which is hard to compute for mVNP, even as a function over the boolean hypercube; a result that perhaps stands on its own.

A proof sketch of Theorem 3.7 can be found in Section 4 and a complete proof can be found in Subsection C.5.

3.3 Monotone circuits with summation and production gates

Note that it is unclear if quantified monotone circuits are closed under compositions. We therefore also consider a model that generalizes quantified monotone circuits and is trivially closed under compositions. Here summation and production gates are allowed to appear anywhere in the circuit.

▶ **Definition 3.8** (Algebraic circuits with summation and production gates). *An* algebraic circuit with summation and production gates *is an algebraic circuit (Definition 2.1) in which the internal nodes can also be summation or production gates (Definition 2.8), in addition to* $+$ *or* $\times$*. A subset of the variables used by the circuit are marked as* auxiliary*. These variables do not appear in the output polynomial(s) of the circuit, and the labels for all the summation and production gates are required to be auxiliary variables.*

The size *of an algebraic circuit with summation and production gates is the number of nodes in the graph.*

An algebraic circuit with summation, production gates is said to be monotone*, if all the constants appearing in it are non-negative.*

We denote by $\mathsf{mVP}_{\mathsf{sum,prod}}$ *the class of all* n*-variate polynomial families of degree* $\mathrm{poly}(n)$ *that are computable by monotone algebraic circuits with summation and production gates of size* $\mathrm{poly}(n)$*.*

Note that even in the non-monotone setting this model is clearly as powerful as quantified circuits, but can be simulated by circuits with projection gates. Again, Malod [18] showed that quantified circuits and circuits with projection gates are equivalent in power. So the class of polynomials efficiently computable by this model is also VPSPACE.

In the monotone setting, however, it is not clear if the power of quantified monotone circuits is the same as that of this model. In particular, we observe the following. Here, we mean "closure under compositions" in a strong sense: if C_1 and C_2 are quantified monotone circuits of size s_1 and s_2 respectively, then the polynomial computed by their composition to have a quantified monotone circuit of size at most $s_1 + s_2$.

▶ **Observation 3.9** (Informal). *Quantified monotone circuits are closed under compositions, if and only if,* $\mathsf{mVP}_{\mathsf{quant}} = \mathsf{mVP}_{\mathsf{sum,prod}}$*.*

Theorem 5.1 gives a formal statement and its proof can be found in Subsection C.3.

We, however, show that even this seemingly stronger model does not help in computing transparent polynomials.

▶ **Theorem 3.10.** *Any monotone algebraic circuit with summation and production gates that computes a transparent polynomial* f*, has size at least* $|\mathrm{supp}(f)|/4$*.*

This shows that transparent polynomials with large support are hard even for this model. A proof sketch can be found in Section 5.

Recall that one way to refute the τ-conjecture for Newton polygons is to show a transparent polynomial in (non-monotone) VP. Theorem 3.10 shows that any transparent polynomial from VP that refutes the conjecture would also witness a separation between VP and a class

potentially much bigger than mVNP[4]. Even though stark separations between monotone and non-monotone models are not unheard of [10, 4], such a result would also be quite interesting and would further highlight the power of subtractions.

3.4 Monotone circuits with projection gates

Finally, adapting the definition of VPSPACE due to Poizat (Definition 2.6) [19], we define monotone circuits with projection gates.

▶ **Definition 3.11** (Monotone algebraic circuits with projection gates). *A* monotone algebraic circuit with projection gates *is an algebraic circuit with projections (Definition 2.6) in which only non-negative constants from the field are allowed to appear as labels of leaves.*

As in Definition 3.8, only the auxiliary *variables can be used as labels for the projection gates. The* size *of a monotone algebraic circuit with projection gates is the number of nodes in the underlying graph.*

We denote by $\mathsf{mVP}_{\mathsf{proj}}$ *the class of all n-variate polynomials of degree* $\mathrm{poly}(n)$ *that are computable by size-*$\mathrm{poly}(n)$ *monotone algebraic circuits with projection gates.*

This model is clearly at least as powerful as monotone circuits with summation and production gates, since $\mathtt{sum}_z = \mathtt{fix}_{(z=0)} + \mathtt{fix}_{(z=1)}$ and $\mathtt{prod}_z = \mathtt{fix}_{(z=0)} \times \mathtt{fix}_{(z=1)}$. It would therefore be interesting to show a separation between the power of the two models.

Even though we are unable to do that, we show that monotone circuits with projection gates are indeed more powerful than quantified monotone circuits, with a $2^{\Omega(\sqrt{m})}$ separation.

▶ **Theorem 3.12.** *The polynomial family* $\{\mathsf{Perm}_n\}$ *can be computed by monotone circuits with projection gates of size $O(n^3)$, but quantified monotone circuits computing it must have size* $2^{\Omega(n)}$.

Finally we show that $\mathsf{mVP}_{\mathsf{proj}}$ is closed under taking homogeneous components.

▶ **Theorem 3.13.** *Suppose f is computed by a size s monotone circuit with projections. Then for any $k \leq \deg(f)$,* $\mathrm{hom}_k(f)$ *has a monotone circuit with projections of size $O(k^2 \cdot s)$.*

Proof sketches of Theorem 3.12 and Theorem 3.13 can be found in Section 6.

3.5 Defining Monotone VPSPACE (mVPSPACE)

We propose the following definition for mVPSPACE.

▶ **Definition 3.14** (Monotone VPSPACE). *A family of polynomials $\{f_n\}$ is said to be in* mVPSPACE *if for all large n, f_n is computable by a monotone algebraic circuit with projection gates (Definition 3.11) of size* $\mathrm{poly}(n)$.

Further if $\{f_n\}$ has degree $\mathrm{poly}(n)$*, then it is said to be in* $\mathsf{mVPSPACE}_b$.

That is, we define $\mathsf{mVPSPACE}_b := \mathsf{mVP}_{\mathsf{proj}}$ and define mVPSPACE along the same lines, but without the restriction on the degree being bounded (since VPSPACE does not impose any restrictions on degree). Some of our reasons for this choice are as follows.

Firstly, being a complexity class, $\mathsf{mVPSPACE}_b$ should be closed under (monotone) affine projections, i.e. setting a few variables to monotone affine polynomials. All of $\mathsf{mVP}_{\mathsf{quant}}$, $\mathsf{mVP}_{\mathsf{sum,prod}}$ and $\mathsf{mVP}_{\mathsf{proj}}$ have this property.

[4] That is, the class of bounded degree polynomials computable by monotone algebraic circuits with summation and production gates.

Further, as mVP and mVNP are closed under taking homogeneous components, it is desirable for a more powerful class to also have this property. Even if $\mathsf{mVP}_{\mathsf{quant}}$ satisfies this, it would not lead to a larger class (Corollary 3.6). Also, it is not clear $\mathsf{mVP}_{\mathsf{sum,prod}}$ is closed under homogenization, while $\mathsf{mVP}_{\mathsf{proj}}$ is (Theorem 3.13).

Finally, we believe that having $\mathsf{Perm}_n \in \mathsf{mVP}_{\mathsf{proj}}$ is an interesting property that further strengthens the case for $\mathsf{mVP}_{\mathsf{proj}}$ being the definition for $\mathsf{mVPSPACE}_b$.

4 Quantified monotone circuits

Computing homogeneous polynomials

▶ **Theorem 3.5.** *Let f be computable by a quantified monotone circuit of size s. If f is homogeneous, then it is expressible as an exponential sum of size at most $O(s \cdot \deg(f))$.*

Proof. Let $d = \deg(f)$, and let $\mathcal{C}$ be a quantified monotone circuit computing f, that uses exactly k production gates. We can then assume that,

$$\mathcal{C}(\mathbf{x}) = \mathtt{sum}_{\mathbf{y}_0}\mathtt{prod}_{z_1}\mathtt{sum}_{\mathbf{y}_1}\mathtt{prod}_{z_2}\cdots\mathtt{sum}_{\mathbf{y}_{k-1}}\mathtt{prod}_{z_k}\mathtt{sum}_{\mathbf{y}_k}g(\mathbf{x},\mathbf{y},\mathbf{z}),$$

without loss of generality, by using some empty $\mathbf{y}_j$s whenever necessary. Note that the $\mathbf{y}_j$s are sets of variables, whereas each of the z_js are single variables.

We now prove the statement in two steps. First, we use the homogeneity of f, and the monotonicity of the quantified circuit, to show that $k \leq \log(d)$.

▷ Claim 4.1. $k \leq \log d$

Proof. For each $i \in [k]$, let $g_i(z_i, \mathbf{x}, \mathbf{w}_i) = \mathtt{sum}_{\mathbf{y}_i}\mathtt{prod}_{z_{i+1}}\mathtt{sum}_{\mathbf{y}_{i+1}}\cdots\mathtt{sum}_{\mathbf{y}_k}g(\mathbf{x},\mathbf{y},\mathbf{z})$. Here $\mathbf{w}_i$ denotes all the auxiliary variables that are alive after "i rounds" of quantifiers. Further, let $h_i(\mathbf{x},\mathbf{w}_i) = \mathtt{prod}_{z_i}g_i(z_i,\mathbf{x},\mathbf{w}_i)$.

Now, $f(\mathbf{x}) = \mathtt{sum}_{\mathbf{y}_0}h_1(\mathbf{x},\mathbf{y}_0)$, and it is homogeneous. Therefore, since h_1 is monotone, it is also homogeneous in $\mathbf{x}$ with degree exactly d. But $\deg_{\mathbf{x}}(h_1) = \deg_{\mathbf{x}}(\mathtt{prod}_{z_1}g_1) = \deg_{\mathbf{x}}(g_1(z_1 = 0)) + \deg_{\mathbf{x}}(g_1(z_1 = 1))$. If we write $g_1(z_1,\mathbf{x},\mathbf{w}_1) = g_{1,0}(\mathbf{x},\mathbf{w}_1) + z \cdot g_{1,1}(z_1,\mathbf{x},\mathbf{w}_1)$, then we have that $g_1(z_1 = 0) = g_{1,0}(\mathbf{x},\mathbf{w}_1)$ and $g_1(z_1 = 1) = g_{1,0}(\mathbf{x},\mathbf{w}_1) + g_{1,1}(z_1 = 1,\mathbf{x},\mathbf{w}_1)$. Since h_1 is homogeneous in $\mathbf{x}$ and g_1 is monotone in all the variables, this must mean that $\deg_{\mathbf{x}}(g_1(z_1 = 0)) = \deg_{\mathbf{x}}(g_1(z_1 = 1)) = \deg_{\mathbf{x}}(g_1) = d/2$. Also, g_1 is homogeneous in $\mathbf{x}$, and thus we can repeat the same argument for h_2, g_2, and so on.

As a result, we see that $\deg(f) = 2^k \cdot \deg_{\mathbf{x}}(g)$, and hence $k \leq \log d$. ◁

We can now make $2^k \leq d$ many copies of the "inner circuit" $g(\mathbf{x},\mathbf{y},\mathbf{z})$, one for each fixing of the $\mathbf{z}$ variables. We then obtain the final exponential sum computing f by using the following "product rule" for summations repeatedly.

$$(\mathtt{sum}_{\mathbf{y}_1}h_1(\mathbf{x},\mathbf{y}_1)) \cdot (\mathtt{sum}_{\mathbf{y}_2}h_2(\mathbf{x},\mathbf{y}_2)) = \mathtt{sum}_{\widetilde{\mathbf{y}}_1,\widetilde{\mathbf{y}}_2}(h_1(\mathbf{x},\widetilde{\mathbf{y}}_1) \cdot h_2(\mathbf{x},\widetilde{\mathbf{y}}_2))$$

Note that in the above case the two summations are over disjoint sets of variables. This can easily be ensured in our case, by treating the $\mathbf{y}$ variables in each of the $2^k \leq d$ copies as mutually disjoint. It is easy to see that the exponential sum has size $O(\mathrm{size}(C), d)$. ◀

▶ Remark 4.2. The first step in the above proof extends more or less as it is, to an arbitrary circuit with summation and production gates. Thus, any circuit with arbitrary summations and productions *that computes a homogeneous polynomial* can be assumed to not contain any production gates, with a polynomial blow-up in size.

However, this does not directly give an efficient exponential sum, because of the second step in the above argument. It crucially uses the fact that for any summation gate g, the number of production gates on a path from g to the root was $O(\log d)$. This ensures that no summation gate (or its auxiliary variable) has to be replicated more than $\mathrm{poly}(d)$ times, which is not necessarily true if we start with an arbitrary circuit with summation gates.

Large exponential sums for arbitrary polynomials

▶ **Theorem 3.7.** *Suppose $f(\mathbf{x})$ is an n-variate, degree-d polynomial computed by a quantified monotone circuit of size s, which uses ℓ summation gates. Then for a set of variables $\mathbf{w}$ of size at most $d \cdot \ell$, there is a monotone circuit $h(\mathbf{x}, \mathbf{w})$ of size at most $d \cdot s$, and a monotone polynomial $A(\mathbf{w})$ such that,*

$$f(\mathbf{x}) = \sum_{\mathbf{b} \in \{0,1\}^{|\mathbf{w}|}} A(\mathbf{w} = \mathbf{b}) \cdot h(\mathbf{x}, \mathbf{w} = \mathbf{b}), \tag{1}$$

where $A(\mathbf{w})$ potentially has circuit size and degree that is exponential in n and ℓ.

We shall need the following simple observation, which follows from the "product-rule" for summations stated earlier.

▶ **Observation 4.3** (Product of exponential sums)**.**

$$\mathtt{prod}_z \mathtt{sum}_{\mathbf{y}} g(\mathbf{x}, \mathbf{y}, z) = \mathtt{sum}_{\mathbf{y}_0, \mathbf{y}_1} \left(g(\mathbf{x}, \mathbf{y}_0, 0) \cdot g(\mathbf{x}, \mathbf{y}_1, 1) \right)$$

Let us see a toy case of trivially moving from a quantified expression to an exponential sum, using Observation 4.3.

$$\begin{aligned}
f(x) &= \mathtt{sum}_{y_1} \mathtt{prod}_{z_1} \mathtt{sum}_{y_2} \mathtt{prod}_{z_2, z_3} \mathtt{sum}_{y_3} g(x, y_1, y_2, y_3, z_1, z_2, z_3) \\
&= \mathtt{sum}_{y_1} \mathtt{prod}_{z_1} \mathtt{sum}_{y_2} \mathtt{prod}_{z_2} \mathtt{sum}_{y_{3,0}, y_{3,1}} \left(\prod_{a_3 \in \{0,1\}} g(x, y_1, y_2, y_{3,a_3}, z_1, z_2, a_3) \right) \\
&= \mathtt{sum}_{y_1} \mathtt{prod}_{z_1} \mathtt{sum}_{y_2, y_{3,(00)}, y_{3,(01)}, y_{3,(10)}, y_{3,(11)}} \left(\prod_{a_2, a_3 \in \{0,1\}} g(\ldots, y_{3,(a_2 a_3)}, z_1, a_2, a_3) \right) \\
&= \mathtt{sum}_{y_1} \mathtt{sum}_{y_{2,*}, y_{3,***}} \left(\prod_{a_1, a_2, a_3 \in \{0,1\}} g(x, y_1, y_{2,a_1}, y_{3,(a_1 a_2 a_3)}, a_1, a_2, a_3) \right)
\end{aligned}$$

In the last line, each $*$ runs over $\{0, 1\}$, so there are $1 + 2 + 8 = 11$ auxiliary variables in total. Note that y_3 has 8 copies, which is due to the 3 production gates "above" the summation gate labelled by it. Similarly, y_2 has just 2 copies, while y_1 has just one. Also, if instead of single auxiliary variables y_2 and y_3 we had sets of auxiliary variables $\mathbf{y}_2$ and $\mathbf{y}_3$, nothing much would change. That is, we would have had 8 copies of the set $\mathbf{y}_3$ and 2 copies of $\mathbf{y}_2$, irrespective of their sizes.

What this shows in general, is that we can trivially move from a quantified expression to an expression which has the form

$$f(\mathbf{x}) = \mathtt{sum}_{\mathbf{Y}} \prod_{\mathbf{a} \in \{0,1\}^r} g_{\mathbf{a}}(\mathbf{x}, \mathbf{y}_{\mathbf{a}})$$

where $\mathbf{Y} = \cup_{\mathbf{a}} \{\mathbf{y}_{\mathbf{a}}\}$, r is the number of production gates in the quantified expression, $|\mathbf{Y}|$ is potentially exponential (since the number of copies of some auxiliary variable might be exponential) but $g_{\mathbf{a}}(\mathbf{x}, \mathbf{y}_{\mathbf{a}}) = g(\mathbf{x}, \mathbf{y} = \mathbf{y}_{\mathbf{a}}, \mathbf{z} = \mathbf{a})$ for a poly-sized circuit $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$.

The key observation that allows us to prove Theorem 3.7 is that if f has degree d, then the number of copies of each auxiliary variable needed in the outer summation gate is at most d. This is because, due to monotonicity, $\deg_{\mathbf{x}}(g_{\mathbf{a}}(\mathbf{x}, \mathbf{y_a})) \neq 0$ for only d many $\mathbf{a} \in \{0,1\}^r$. A complete proof of Theorem 3.7 can be found in Subsection C.5.

5 Monotone circuits with summation and production gates

In this section, we give the proof overview of Theorem 3.10.

▶ **Theorem 3.10.** *Any monotone algebraic circuit with summation and production gates that computes a transparent polynomial f, has size at least* $|\mathrm{supp}(f)|/4$.

This result is an extension of the ideas in the work of Hrubeš & Yehudayoff [11]. Their argument shows that any bivariate monotone circuit of size s that computes a polynomial with *convexly independent support* outputs a polynomial with support at most $4s$. They achieve this by keeping track of the largest polygon (in terms of the number of vertices) that one can build using the polynomials computed at all the gates in the circuit. They then inductively show that no gate (leaf, addition, multiplication) can increase the number of vertices by 4. We are able to show the same bound for production and summation gates, by working with a monotone bivariate circuit over y_1, y_2 that is allowed some auxiliary variables z for summations and productions.

An important component of the proof in [11] is that if the sum or product of two monotone polynomials is convexly independent, then so are each of the two inputs. However, allowing for summations and productions means that some monomials that are computed internally could get "zeroed out". In fact, summation and production gates do not quite "preserve convex dependencies". For example, the convexly dependent support $\{y_1y_2, y_1y_2z, y_1y_2z^2\}$ when passed through $\mathtt{sum}_z$ produces just $\{y_1y_2\}$, which is convexly independent.

In order to prove Theorem 3.10, one can get around this by working directly with the support projected down to the "true" variables, which we call $\mathbf{y}$-support in our arguments. It turns out that summations and productions indeed preserve convex dependencies that are in the $\mathbf{y}$ support of the input polynomial. Since the proof follows exactly along the same lines as the one in [11], we omit the proof here. A formal proof can be found in the full version [2].

Quantified monotone circuits and compositions

▶ **Observation 3.9** (Informal). *Quantified monotone circuits are closed under compositions, if and only if,* $\mathsf{mVP}_{\mathsf{quant}} = \mathsf{mVP}_{\mathsf{sum,prod}}$.

Even though this statement appears to be straightforward, formally stating it requires a bit more care. In particular, we require quantified circuits to be closed under compositions in a strong sense, similar to usual algebraic circuits. Doing that yields the following theorem, which we prove in Subsection C.3.

▶ **Theorem 5.1.** *Suppose that for any* multi-output *quantified monotone circuit $\mathcal{C}$ of size s with r inputs, and any* multi-output *quantified monotone circuit $\mathcal{C}'$ of size s' with r outputs, we have that the polynomial computed by $\mathcal{C} \circ \mathcal{C}'$ has a quantified monotone circuit of size at most $(s + s')$.*

Then, any multi-output*, monotone circuit with summation and production gates of size $\tilde{s}$ can be simulated by a* multi-output *quantified monotone circuit of size at most $\tilde{s}$, and hence* $\mathsf{mVP}_{\mathsf{quant}} = \mathsf{mVP}_{\mathsf{sum,prod}}$.

The converse is also true.

6 Monotone circuits with projection gates

Exponential separation from quantified circuits

▶ **Theorem 3.12.** *The polynomial family* $\{\mathsf{Perm}_n\}$ *can be computed by monotone circuits with projection gates of size* $O(n^3)$*, but quantified monotone circuits computing it must have size* $2^{\Omega(n)}$*.*

We begin by proving that $\mathsf{Perm}_n \in \mathsf{mVP}_{\mathsf{proj}}$.

▶ **Theorem 6.1.** *There is a monotone circuit with projection gates of size* $O(n^3)$ *that computes* Perm_n*.*

Proof. We first define a polynomial P_0 such that all its monomials contain exactly one $\mathbf{x}$-variable from each row.

$$\text{Let } P_0(\mathbf{x},\mathbf{y}) := \left(\sum_{j=1}^{n} y_{1,j}x_{1,j}\right)\left(\sum_{j=1}^{n} y_{2,j}x_{2,j}\right)\cdots\left(\sum_{j=1}^{n} y_{n,j}x_{n,j}\right).$$

Note that P_0 has n^2 auxiliary variables $\mathbf{y}$, one attached to each "true" variable $x_{i,j}$. We now want to use these to progressively prune the monomials that pick up multiple variables from the jth column by projecting the n variables $y_{1,j},\ldots,y_{n,j}$.

Let $e_1,\ldots,e_n \in \{0,1\}^n$ be such that $e_i(k) = 1 \Leftrightarrow i = k$, and define for each $j \in [n]$,

$$P_j := \sum_{i\in[n]} \mathtt{fix}_{(y_{1,j}=e_i(1))}\left(\mathtt{fix}_{(y_{2,j}=e_i(2))}\left(\cdots\left(\mathtt{fix}_{(y_{n,j}=e_i(n))}\left(P_{j-1}\right)\right)\right)\right). \tag{2}$$

The following claim is now easy to verify.

▷ Claim 6.2. For all $j \in [n]$, P_j contains all the monomials from P_{j-1} that are supported on exactly one $\mathbf{x}$-variable from the jth column.

As a result, the monomials in P_n are exactly those of the monomials in Perm_n. Additionally, for each j, the auxiliary variables in P_j are only from the columns $j+1,\ldots,n$; thus $P_n = \mathsf{Perm}_n$.

The size of our circuit is $O(n^3)$, since $\mathrm{size}(P_0) = O(n^2)$ and $\mathrm{size}(P_j) = \mathrm{size}(P_{j-1}) + O(n^2)$. This proves Theorem 6.1. ◀

▶ Remark 6.3. Our upper bound above also implies that any polynomial (family) that can be expressed as the permanent of a monotone matrix of size $\mathrm{poly}(n)$ (called monotone p-projection of Perm_n) can also be computed by efficient monotone circuits with projection gates. Although Perm_n is complete for non-monotone VNP, it is *not* the case that all monotone polynomials in VNP are monotone p-projections of Perm_n, as shown by Grochow [8].

The proof of Theorem 3.12 now follows from the following simple extension of an observation due to Yehudayoff [26] and the classical lower bound of Jerrum & Snir [12] against monotone algebraic circuits for Perm_n. A complete proof can be found in Subsection C.1.

▶ **Lemma 6.4.** *Let* $f(\mathbf{x})$ *be a monotone polynomial whose support cannot be written as a non-trivial product of two sets, and for some monotone polynomial* $g(\mathbf{x},\mathbf{z})$*, suppose we have* $f(\mathbf{x}) = \mathtt{Q}^{(1)}_{z_1}\mathtt{Q}^{(2)}_{z_2}\cdots\mathtt{Q}^{(m)}_{z_m} g(\mathbf{x},\mathbf{z})$ *with* $\mathtt{Q}^{(i)} \in \{\mathtt{sum},\mathtt{prod}\}$ *for each* $i \in [m]$*.*

Then $\mathrm{supp}(f(\mathbf{x})) = \mathrm{supp}(g(\mathbf{x},\bar{1}))$*.*

Closure under homogenization

▶ **Theorem 3.13.** *Suppose f is computed by a size s monotone circuit with projections. Then for any $k \leq \deg(f)$, $\hom_k(f)$ has a monotone circuit with projections of size $O(k^2 \cdot s)$.*

We show this using the classical argument of "gate replication" and the complete proof can be found in Subsection C.2.

7 Conclusion

Our work is an attempt at understanding the hardness of transparent polynomials for monotone algebraic models. We observe that the lower bound of Hrubeš & Yehudayoff [11] extends beyond monotone VNP, and therefore turn to exploring the class VPSPACE from the non-monotone world. This exploration reveals that the natural monotone analogues of the multiple equivalent definitions of VPSPACE have contrasting powers. Additionally, transparent polynomials turn out to be as hard for some of these analogues as they are for usual monotone circuits. The following are some interesting open threads from our work.

- We suspect that transparency is a highly restrictive property, especially for monotone computation. Therefore, we conjecture that if f is a transparent polynomial being computed by a size-s monotone circuit with projection gates, then $|\mathrm{supp}(f)| \leq 2^{\mathrm{polylog}(s)}$. It would be interesting (at least to us) to see a proof or a refutation of this conjecture. An immediate hurdle in extending the techniques in [11] (as in Theorem 3.10) to mVPSPACE, is that unlike summations and productions, 0-projections do not preserve convex dependencies, even if we restrict to the "true" variables.
- Along similar lines, a possibly simpler goal is to show a non-monotone circuit upper bound for a transparent polynomial. Since transparency only restricts the support of the polynomial, one is free to choose any real coefficients to ensure that it is in VP. In particular, this brings powerful non-monotone tricks like interpolation into play. Among other things, such a result would refute the notoriously open τ-conjecture for Newton polygons.
- Another question we would like to highlight is separating mVNP and quantified monotone circuits. As mentioned in the discussion following Theorem 3.7, such a separation would yield a (high degree) polynomial that is hard for mVNP even as a function over the boolean hypercube. Such a polynomial might be of interest, perhaps, even in the non-monotone setting.

References

1. Bruno Pasqualotto Cavalar, Mrinal Kumar, and Benjamin Rossman. Monotone circuit lower bounds from robust sunflowers. In Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, editors, *LATIN 2020: Theoretical Informatics – 14th Latin American Symposium, São Paulo, Brazil, January 5-8, 2021, Proceedings*, volume 12118 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2020. doi:10.1007/978-3-030-61792-9_25.
2. Prerona Chatterjee, Kshitij Gajjar, and Anamay Tengse. Monotone classes beyond VNP. *CoRR*, abs/2202.13103, 2022. arXiv:2202.13103.
3. Arkadev Chattopadhyay, Rajit Datta, Utsab Ghosal, and Partha Mukhopadhyay. Monotone complexity of spanning tree polynomial re-visited. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 – February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 39:1–39:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.39.

4 Arkadev Chattopadhyay, Rajit Datta, and Partha Mukhopadhyay. Lower bounds for monotone arithmetic circuits via communication complexity. 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 786–799. ACM, 2021. doi:10.1145/3406325.3451069.

5 Arkadev Chattopadhyay, Utsab Ghosal, and Partha Mukhopadhyay. Robustly separating the arithmetic monotone hierarchy via graph inner-product. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 12:1–12:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.12.

6 S. B. Gashkov and I. S. Sergeev. A method for deriving lower bounds for the complexity of monotone arithmetic circuits computing real polynomials. *Sbornik. Mathematics*, 203(10), 2012.

7 S.B. Gashkov. The complexity of monotone computations of polynomials. *Moscow University Math Bulletin*, 42(5):1–8, 1987.

8 Joshua A. Grochow. Monotone projection lower bounds from extended formulation lower bounds. *Theory of Computing*, 13(18):1–15, 2017. doi:10.4086/toc.2017.v013a018.

9 Pavel Hrubes and Amir Yehudayoff. On isoperimetric profiles and computational complexity. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 89:1–89:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.89.

10 Pavel Hrubeš and Amir Yehudayoff. Formulas are exponentially stronger than monotone circuits in non-commutative setting. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 10–14. IEEE Computer Society, 2013. doi:10.1109/CCC.2013.11.

11 Pavel Hrubeš and Amir Yehudayoff. Shadows of newton polytopes. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.9.

12 Mark Jerrum and Marc Snir. Some exact complexity results for straight-line computations over semirings. *Journal of the ACM*, 29(3):874–897, 1982. doi:10.1145/322326.322341.

13 O. M. Kasim-Zade. Arithmetic complexity of monotone polynomials. *Theoretical Problems in Cybernetics. Abstracts of lectures*, pages 68–69, 1986.

14 Pascal Koiran and Sylvain Perifel. VPSPACE and a transfer theorem over the reals. *Computational Complexity*, 18(4):551–575, 2009. doi:10.1007/s00037-009-0269-1.

15 Pascal Koiran and Sylvain Perifel. Vpspace and a transfer theorem over the complex field. *Theoretical Computer Science*, 410(50):5244–5251, 2009. Mathematical Foundations of Computer Science (MFCS 2007). doi:10.1016/j.tcs.2009.08.026.

16 Pascal Koiran, Natacha Portier, Sébastien Tavenas, and Stéphan Thomassé. A τ-conjecture for newton polygons. *Foundations of Computational Mathematics*, 15:185–197, 2015. doi:10.1007/s10208-014-9216-x.

17 Meena Mahajan and B. V. Raghavendra Rao. Small space analogues of valiant's classes and the limitations of skew formulas. *Computational Complexity*, 22(1):1–38, 2013. doi:10.1007/s00037-011-0024-2.

18 Guillaume Malod. Succinct algebraic branching programs characterizing non-uniform complexity classes. In *Fundamentals of Computation Theory – 18th International Symposium, FCT 2011, Oslo, Norway, August 22-25, 2011. Proceedings*, pages 205–216, 2011. doi:10.1007/978-3-642-22953-4_18.

19 Bruno Poizat. A la recherche de la definition de la complexite d'espace pour le calcul des polynomes a la maniere de valiant. *J. Symb. Log.*, 73(4):1179–1201, 2008. doi:10.2178/jsl/1230396913.

20 Ran Raz and Amir Yehudayoff. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. *Journal of Computer and System Sciences*, 77(1):167–190, 2011. doi:10.1016/j.jcss.2010.06.013.

21 Claus-Peter Schnorr. A lower bound on the number of additions in monotone computations. *Theor. Comput. Sci.*, 2(3):305–315, 1976. doi:10.1016/0304-3975(76)90083-9.

22 Eli Shamir and Marc Snir. *Lower bounds on the number of multiplications and the number of additions in monotone computations.* IBM Thomas J. Watson Research Division, 1977.

23 Eli Shamir and Marc Snir. On the depth complexity of formulas. *Math. Syst. Theory*, 13:301–322, 1980. doi:10.1007/BF01744302.

24 Srikanth Srinivasan. Strongly exponential separation between monotone VP and monotone VNP. *ACM Transactions on Computing Theory*, 12(4):23:1–23:12, 2020. doi:10.1145/3417758.

25 Leslie G. Valiant. Negation can be exponentially powerful. *Theor. Comput. Sci.*, 12:303–314, 1980. doi:10.1016/0304-3975(80)90060-2.

26 Amir Yehudayoff. Separating monotone VP and VNP. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 425–429. ACM, 2019. doi:10.1145/3313276.3316311.

A Deferred Formal Definitions and Statements

▶ **Definition A.1** (Newton polytopes). *For a polynomial $f(\mathbf{x})$, its Newton polytope* $\mathsf{Newt}(f) \subseteq \mathbb{R}^n$*, is defined as the convex hull of the* exponent vectors *of the monomials in its support.*

$$\mathsf{Newt}(f) := \operatorname{conv}\left(\{\mathbf{e} : \mathbf{x}^{\mathbf{e}} \in \operatorname{supp}(f)\}\right)$$

A point $\mathbf{e} \in \mathsf{Newt}(f)$ *is said to be a* vertex*, if it cannot be written as a convex combination of* other *points in* $\mathsf{Newt}(f)$*. We denote the set of all vertices of a polytope* $\mathcal{P}$ *using* $\operatorname{vert}(\mathcal{P})$*.*

▶ **Conjecture A.2** (τ conjecture for Newton polytopes [16]). *Suppose $f(x, y)$ is a bivariate polynomial that can be written as $\sum_{i\in[s]} \prod_{j\in[r]} T_{i,j}(x, y)$, where each $T_{i,j}$ has sparsity at most p. Then the Newton polygon of f has* $\operatorname{poly}(s, r, p)$ *vertices.*

B Definitions of VPSPACE relying on boolean computation

In this section we briefly address why we did not study monotone analogues of the definitions due to Koiran & Perifel [15, 14], and Mahajan & Rao [17].

Koiran & Perifel define uniform VPSPACE as the class of families $\{f_n\}$ of $\operatorname{poly}(n)$-variate polynomials of degree at most $2^{\operatorname{poly}(n)}$, such that there is a PSPACE machine that computes the *coefficient function* of $\{f_n\}$. Here, the coefficient function of $\{f_n\}$ can be seen to map a pair $(1^n, \mathbf{e})$ to the coefficient of $\mathbf{x}^{\mathbf{e}}$ in f_n.

Non-uniform VPSPACE is then defined by replacing PSPACE by its non-uniform analogue, PSPACE/poly. Since there are no monotone analogues of Turing machines, perhaps the only possible monotone analogue of this definition is to insist on the coefficient function being monotone, which results in an absurdly weak class (the "largest" monomial will always be present).

Mahajan & Rao [17] look at the notion of *width* of a circuit – all gates are assigned heights, such that the height of any gate is *exactly* one larger than the height of its highest child. The width of the circuit is the maximum number of nodes that have the same height. They then define $\mathsf{VSPACE}(S(n))$, as the class of families that are computable by circuits of width $S(n)$ and size at most $\max\left\{2^{S(n)}, \mathrm{poly}(n)\right\}$.

The class uniform $\mathsf{VSPACE}(S(n))$ further requires that the circuits be $\mathsf{DSPACE}(\mathsf{S}(\mathsf{n}))$-uniform. Although their non-uniform definition is purely algebraic, it is a bit unnatural for space $S(n) \gg \log n$ (as also pointed out in their paper), since such circuits may not even have a $\mathrm{poly}(n)$-sized description. We therefore do not analyse a monotone analogue for their definition.

C Deferred Proofs

C.1 Proof of Theorem 3.12

We first consider Lemma 6.4, which is a simple extension of the following observation due to Yehudayoff [26].

▶ **Observation C.1** ([26]). *Let $g(\mathbf{x}, z)$ be a monotone polynomial and let $c > 0$. Then for any monomial $\mathbf{x}^{\mathbf{e}} z^j$ in the support of g, $\mathbf{x}^{\mathbf{e}} \in \mathrm{supp}(g(\mathbf{x}, z = c))$.*

We now formally prove Lemma 6.4. For sets of monomials A and B, their product is defined as $A \times B = \{m \cdot m' : m \in A, m' \in B\}$; a non-trivial product is when neither A nor B is just $\{1\}$.

▶ **Lemma 6.4.** *Let $f(\mathbf{x})$ be a monotone polynomial whose support cannot be written as a non-trivial product of two sets, and for some monotone polynomial $g(\mathbf{x}, \mathbf{z})$, suppose we have $f(\mathbf{x}) = \mathtt{Q}^{(1)}_{z_1} \mathtt{Q}^{(2)}_{z_2} \cdots \mathtt{Q}^{(m)}_{z_m} g(\mathbf{x}, \mathbf{z})$ with $\mathtt{Q}^{(i)} \in \{\mathtt{sum}, \mathtt{prod}\}$ for each $i \in [m]$.*

Then $\mathrm{supp}(f(\mathbf{x})) = \mathrm{supp}(g(\mathbf{x}, \bar{1}))$.

Proof. By Observation C.1, it is enough to show the statement of the lemma for $m = 1$. Therefore, suppose $f(\mathbf{x}) = \mathtt{sum}_z g(\mathbf{x}, z)$. Then $f(\mathbf{x}) = g(\mathbf{x}, 0) + g(\mathbf{x}, 1)$, and hence $\mathrm{supp}(f) = \mathrm{supp}(g(\mathbf{x}, 1))$, since g is monotone.

Next, $f(\mathbf{x}) = \prod_z g(\mathbf{x}, z)$ means that $f(\mathbf{x}) = g(\mathbf{x}, 0) \cdot g(\mathbf{x}, 1)$. As $\mathrm{supp}(f)$ cannot be written as a non-trivial product of two sets, and since g is monotone, this must mean that $g(\mathbf{x}, 0)$ is a constant and $\mathrm{supp}(f(\mathbf{x})) = \mathrm{supp}(g(\mathbf{x}, 1))$ as claimed. ◀

Finally, let us complete the proof of Theorem 3.12.

▶ **Theorem 3.12.** *The polynomial family $\{\mathsf{Perm}_n\}$ can be computed by monotone circuits with projection gates of size $O(n^3)$, but quantified monotone circuits computing it must have size $2^{\Omega(n)}$.*

Proof. Let us assume that there is a quantified monotone circuit of size s computing Perm_n. Then,

$$\mathsf{Perm}_n(\mathbf{x}) = \mathtt{Q}^{(1)}_{z_1} \mathtt{Q}^{(2)}_{z_2} \cdots \mathtt{Q}^{(m)}_{z_m} f(\mathbf{x}, \mathbf{z})$$

for some $m \leq s$ and $\mathtt{Q}^{(i)} \in \{\mathtt{sum}, \mathtt{prod}\}$ for each $i \in [m]$.

Note that, by definition, $f(\mathbf{x}, \mathbf{z})$ is computable by a monotone algebraic circuit of size at most s and therefore $f(\mathbf{x}, \bar{1})$ is computable by a monotone algebraic circuit of size at most s. On the other hand, by Lemma 6.4, the support of $f(\mathbf{x}, \bar{1})$ is the same as that of Perm_n since Perm_n is irreducible. The required lower bound now follows from the fact that the $2^{\Omega(n)}$ lower bound proved by Jerrum & Snir [12] for Perm_n against monotone algebraic circuits, works for any polynomial that has support equal to the support of Perm_n. ◀

C.2 Proof of Theorem 3.13

▶ **Theorem 3.13.** *Suppose f is computed by a size s monotone circuit with projections. Then for any $k \leq \deg(f)$, $\hom_k(f)$ has a monotone circuit with projections of size $O(k^2 \cdot s)$.*

Proof. We show this using the classical argument of "gate replication". Given a circuit $\mathcal{C}$, we construct another circuit $\mathcal{C}'$ that has $(k+1)$ copies of each gate in $\mathcal{C}$. For a gate $g \in \mathcal{C}$, the corresponding gates $g_0, g_1, \ldots, g_k$ shall compute $\hom_i([g])$ for each $i \leq k$, where $[g]$ is the polynomial computed at g. Here and throughout the proof, the degree of a polynomial always refers to its degree in the $\mathbf{x}$-variables.

The following can now be easily checked, using the fact that $[g]$ is always a monotone polynomial.

- If $[g]$ is a leaf labelled with a "true" variable x_i, then $[g_1] = x_i$ and $[g_i] = 0$ for all other i.
- If $[g]$ is any other leaf, then $[g_0] = [g]$ and $[g_i] = 0$ for all other i.
- If $[g] = [u] + [v]$, then $[g_i] = [u_i] + [v_i]$ for all i.
- If $[g] = \texttt{fix}_{(z=b)}[u]$, then $[g_i] = \hom_i([g]) = \texttt{fix}_{(z=b)} \hom_i([u]) = \texttt{fix}_{(z=b)}[u_i]$.
- If $[g] = [u] \times [v]$, then $[g_i] = \sum_{j \leq i} [u_j] \times [v_{i-j}]$, for each i.

The last case incurs the largest blow-up in size, which adds $O(k^2)$ many gates in $\mathcal{C}'$ for one gate in $\mathcal{C}$. This finishes the proof. ◀

C.3 Proof of Theorem 5.1

▶ **Theorem 5.1.** *Suppose that for any* multi-output *quantified monotone circuit $\mathcal{C}$ of size s with r inputs, and any* multi-output *quantified monotone circuit $\mathcal{C}'$ of size s' with r outputs, we have that the polynomial computed by $\mathcal{C} \circ \mathcal{C}'$ has a quantified monotone circuit of size at most $(s + s')$.*

Then, any multi-output*, monotone circuit with summation and production gates of size $\tilde{s}$ can be simulated by a* multi-output *quantified monotone circuit of size at most $\tilde{s}$, and hence* $\mathsf{mVP}_{\mathsf{quant}} = \mathsf{mVP}_{\mathsf{sum,prod}}$.

The converse is also true.

Proof. One direction of the implication is clearly true because circuits with (arbitrary) summation and production gates have the stated property by definition.

For the converse, let us assume that quantified monotone circuits have the property. We show that this implies that the two models in question have the same power.

Consider a circuit $\mathcal{C}$ of size s with summation and production gates. We group the gates in $\mathcal{C}$ in "bands" numbered from the bottom to the top, in the following way.

- The 0-th band consists only of leaves
- Odd bands consist only of addition or multiplication gates.
- Even bands (other than 0) only consist of summation or production gates.
- The gates in band i can have edges incoming from only bands $j \leq i$.

Now, given a circuit $\tilde{\mathcal{C}}$ of size $\tilde{s}$ with summation and production gates, we express it as a quantified monotone circuit of size $O(s)$ by inducting on the number of bands in it.

For the base case, when $\tilde{\mathcal{C}}$ has up to two bands, it is already a quantified monotone circuit.

In general, if $\tilde{\mathcal{C}}$ has $2b'$ bands, we look at the circuit formed by bands $2b'$ and $(2b'-1)$ as a quantified monotone circuit; let its size be s. By induction, the *multi-output* circuit formed by the bands 0 to $2b'-2$ can be expressed as a multi-output, quantified monotone circuit of size at most $s' = \tilde{s} - s$, call it $\mathcal{C}'$. Now from the hypothesis, the composition $\mathcal{C} \circ \mathcal{C}'$ is also computable by a quantified monotone circuit of size at most $s + s' \leq \tilde{s}$. ◀

C.4 Proof of Theorem 3.2

▶ **Theorem 3.2.** *If a polynomial family $\{f_n\}$ of degree $\mathrm{poly}(n)$ is computable by monotone succinct ABPs of complexity $\mathrm{poly}(n)$, then $\{f_n\} \in \mathsf{mVNP}$.*

Proof. Let $\mathcal{A} = (B, \mathbf{s}, \mathbf{t}, \ell)$ be the monotone succinct ABP computing f, with $|\mathbf{s}| = |\mathbf{t}| = r$. Then we observe the following.

▷ Claim C.2. If $\ell > 1$, then $\ell \leq \deg(f) + 2$.

Proof. Let $\beta(\mathbf{u}, \mathbf{v}, \mathbf{x})$ be the *monotone* $(2r + n)$-variate polynomial computed by the circuit B. Due to B being monotone, for any $\mathbf{e} \in \mathbb{N}^n$ we have that if the monomial $\mathbf{x}^{\mathbf{e}}$ appears in any edge-label $(\mathbf{a}, \mathbf{b})$, then it also appears in the label of $(\bar{1}, \bar{1})$. Therefore, $\deg_{\mathbf{x}}(\beta(\mathbf{a}, \mathbf{b}, \mathbf{x})) \leq \deg_{\mathbf{x}}(\beta(\bar{1}, \bar{1}, \mathbf{x}))$ for all $\mathbf{a}, \mathbf{b}$. Similarly, $\deg_{\mathbf{x}}(\beta(\mathbf{s}, \mathbf{b}, \mathbf{x})) \leq \deg_{\mathbf{x}}(\beta(\mathbf{s}, \bar{1}, \mathbf{x}))$ and $\deg_{\mathbf{x}}(\beta(\mathbf{a}, \mathbf{t}, \mathbf{x})) \leq \deg_{\mathbf{x}}(\beta(\bar{1}, \mathbf{t}, \mathbf{x}))$ for all $\mathbf{a}, \mathbf{b}$. This shows that if $\ell > 1$, then

$$\deg(f) = \deg(\beta(\mathbf{s}, \bar{1}, \mathbf{x}) \cdot \beta(\bar{1}, \bar{1}, \mathbf{x})^{\ell-2} \cdot \beta(\bar{1}, \mathbf{t}, \mathbf{x})) \geq \ell - 2. \qquad \triangleleft$$

As a result of the above claim, for $d = \deg(f)$, we have the following.

$$\begin{aligned} f(\mathbf{x}) &= \beta(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{j=1}^{d+1} (\text{sum of } \mathbf{s}\text{–}\mathbf{t} \text{ paths through } j \text{ intermediate vertices}) \\ &= \beta(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{j=1}^{d+1} \left(\sum_{\mathbf{a}_1, \ldots, \mathbf{a}_j \in \{0,1\}^r} \beta(\mathbf{s}, \mathbf{a}_1, \mathbf{x}) \cdot \left(\prod_{k=1}^{j-1} \beta(\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{x}) \right) \cdot \beta(\mathbf{a}_j, \mathbf{t}, \mathbf{x}) \right) \\ &= \beta(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \\ &\qquad \sum_{\mathbf{a}_1, \ldots, \mathbf{a}_{d+1} \in \{0,1\}^r} \sum_{j=1}^{d+1} 2^{-r(d+1-j)} \left(\beta(\mathbf{s}, \mathbf{a}_1, \mathbf{x}) \cdot \left(\prod_{k=1}^{j-1} \beta(\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{x}) \right) \cdot \beta(\mathbf{a}_j, \mathbf{t}, \mathbf{x}) \right). \end{aligned}$$

This can be rewritten as follows.

$$\sum_{\mathbf{a}_1, \ldots, \mathbf{a}_{d+1}} \left(2^{-r(d+1)} \beta(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{j=1}^{d+1} 2^{-r(d+1-j)} \beta(\mathbf{s}, \mathbf{a}_1, \mathbf{x}) \left(\prod_{k=1}^{j-1} \beta(\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{x}) \right) \beta(\mathbf{a}_j, \mathbf{t}, \mathbf{x}) \right)$$

This is clearly a poly-sized exponential sum as $d = \mathrm{poly}(n)$ and B is a monotone circuit of size $\mathrm{poly}(n)$. ◀

C.5 Proof of Theorem 3.7

We introduce a new shorthand for this section. For a vector $\mathbf{a} = \{a_1, a_2, \ldots, a_\ell\}$ and a number $k \leq \ell$, we use $\mathbf{a}[: k]$ to denote the *prefix* vector $\{a_1, a_2, \ldots, a_k\}$. With this new notation, we can express the last line of our toy example in Section 4 is as follows.

$$f(x) = \mathtt{sum}_{y_1} \mathtt{sum}_{y_{2,*}, y_{3,***}} \left(\prod_{\mathbf{a} \in \{0,1\}^3} g(x, y_1, y_{2,\mathbf{a}[:1]}, y_{3,\mathbf{a}[:3]}, a_1, a_2, a_3) \right)$$

We are now ready to prove Theorem 3.7. We start by recalling the statement of the theorem.

▶ **Theorem 3.7.** *Suppose $f(\mathbf{x})$ is an n-variate, degree-d polynomial computed by a quantified monotone circuit of size s, which uses ℓ summation gates. Then for a set of variables $\mathbf{w}$ of size at most $d \cdot \ell$, there is a monotone circuit $h(\mathbf{x}, \mathbf{w})$ of size at most $d \cdot s$, and a monotone polynomial $A(\mathbf{w})$ such that,*

$$f(\mathbf{x}) = \sum_{\mathbf{b} \in \{0,1\}^{|\mathbf{w}|}} A(\mathbf{w} = \mathbf{b}) \cdot h(\mathbf{x}, \mathbf{w} = \mathbf{b}), \tag{1}$$

where $A(\mathbf{w})$ potentially has circuit size and degree that is exponential in n and ℓ.

Proof. The first step is to obtain a trivial exponential sum for the quantified expression, as in the discussion above.

▷ Claim C.3. Suppose $f(\mathbf{x})$ can be expressed as the following quantified circuit.

$$f(\mathbf{x}) = \mathtt{sum}_{\mathbf{y}_1} \mathtt{prod}_{\mathbf{z}_1} \mathtt{sum}_{\mathbf{y}_2} \mathtt{prod}_{\mathbf{z}_2} \cdots \mathtt{prod}_{\mathbf{z}_k} \mathtt{sum}_{\mathbf{y}_{k+1}} g(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_{k+1}, \mathbf{z}_1, \ldots, \mathbf{z}_k)$$

Let $m_i = |\mathbf{z}_i|$, and further let $M_i = m_1 + m_2 + \cdots + m_i$, for each $i \in [k]$. Also, let $\mathbf{y} = \mathbf{y}_1 \cup \mathbf{y}_2 \cup \cdots \cup \mathbf{y}_{k+1}$, and $\mathbf{z} = \mathbf{z}_1 \cup \mathbf{z}_2 \cup \cdots \cup \mathbf{z}_k$

Then $f(\mathbf{x})$ can also be expressed as the following exponential sum.

$$f(\mathbf{x}) = \mathtt{sum}_{\mathbf{Y}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_{2,\mathbf{a}[:M_1]}, \mathbf{y}_{3,\mathbf{a}[:M_2]}, \ldots, \mathbf{y}_{k+1,\mathbf{a}[:M_k]}, \mathbf{z} = \mathbf{a}) \right)$$

Here $\mathbf{Y}$ is a set of all y-variables, of size $\left(1 + \sum_i 2^{M_i}\right)$ that is defined as follows.

$$\mathbf{Y} = \bigcup_{\mathbf{a} \in \{0,1\}^{M_k}} (\mathbf{y}_1 \cup \mathbf{y}_{2,\mathbf{a}[:M1]} \cup \cdots \cup \mathbf{y}_{k+1,\mathbf{a}[:M_k]}) \qquad \triangleleft$$

Even though the claim is fairly verbose, it is easy to verify given the discussion before the lemma, so we will not explicitly prove it.

As the next step, we shall use the fact that the "inner circuit" g is monotone, to bound the degree of f from below.

$$\begin{aligned}
\deg(f) &= \deg_{\mathbf{x}} \left(\mathtt{sum}_{\mathbf{Y}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_{2,\mathbf{a}[:M_1]}, \ldots, \mathbf{y}_{k+1,\mathbf{a}[:M_k]}, \mathbf{z} = \mathbf{a}) \right) \right) \\
(g \text{ is monotone}) &= \deg_{\mathbf{x}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{1}, \mathbf{z} = \mathbf{a}) \right) \\
&\geq \sum_{\mathbf{a} \in \{0,1\}^{M_k}} \deg(g(\mathbf{x}, \mathbf{1}, \mathbf{z} = \mathbf{a}))
\end{aligned}$$

Therefore, since f has degree $d = \deg(f)$, it must be the case that for all but d fixings $\mathbf{a}$ of $\mathbf{z}$, $g(\mathbf{x}, \mathbf{y}, \mathbf{a})$ is a constant in terms of $\mathbf{x}$ for any $\{0,1\}$-assignment to the variables in $\mathbf{y}$.

Let $\mathcal{A} := \left\{ \mathbf{a} \in \{0,1\}^{M_k} : \deg_{\mathbf{x}} (g(\mathbf{x}, \mathbf{b}, \mathbf{a})) > 0 \text{ for some } \mathbf{b} \in \{0,1\}^{|\mathbf{y}|} \right\}$, and let $\mathcal{A}_0 := \{0,1\}^{M_k} \setminus \mathcal{A}$. We therefore have that $|\mathcal{A}| \leq d$. Further, let $\mathbf{Y}_1 := \bigcup_{\mathbf{a} \in \mathcal{A}} (\mathbf{y}_1 \cup \mathbf{y}_{2,\mathbf{a}[:M1]} \cup \cdots \cup \mathbf{y}_{k+1,\mathbf{a}[:M_k]})$, and let $\mathbf{Y}_0 := \mathbf{Y} \setminus \mathbf{Y}_1$. Note that now $|\mathbf{Y}_1| \leq |\mathcal{A}| \cdot |\mathbf{y}| \leq d \cdot m$.

We can now simplify the exponential sum in Claim C.3 and finish the proof as follows, where $\mathbf{y_a}$ refers to $(\mathbf{y}_1, \mathbf{y}_{2,\mathbf{a}[:M_1]}, \cdots, \mathbf{y}_{k+1,\mathbf{a}[:M_k]})$.

$$\begin{aligned}
f(\mathbf{x}) &= \mathtt{sum}_{\mathbf{Y}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \\
\text{(for appropriate } \mathbf{y_a}) &= \mathtt{sum}_{\mathbf{Y}} \left(\left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \\
\text{(first term “}\mathbf{x}\text{-free”)} &= \mathtt{sum}_{\mathbf{Y}} \left(\left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{0}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \\
&= \mathtt{sum}_{\mathbf{Y}_1, \mathbf{Y}_0} \left(\left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{0}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \\
\text{(regroup terms)} &= \mathtt{sum}_{\mathbf{Y}_1} \left(\mathtt{sum}_{\mathbf{Y}_0} \left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{0}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \\
\text{(simplify)} &= \mathtt{sum}_{\mathbf{Y}_1} A(\mathbf{Y}_1) \cdot h(\mathbf{x}, \mathbf{Y}_1)
\end{aligned}$$

As claimed, the size of h is at most $|\mathcal{A}| \cdot \text{size}(g) \leq d \cdot s$, while $A(\mathbf{Y}_1)$ is a fairly structured polynomial despite its exponential size and degree. ◀

Approximate Maximum Rank Aggregation: Beyond the Worst-Case

Yan Hong Yao Alvin
National University of Singapore, Singapore

Diptarka Chakraborty
National University of Singapore, Singapore

Abstract

The fundamental task of *rank aggregation* is to combine multiple rankings on a group of candidates into a single ranking to mitigate biases inherent in individual input rankings. This task has a myriad of applications, such as in social choice theory, collaborative filtering, web search, statistics, databases, sports, and admission systems. One popular version of this task, *maximum rank aggregation* (or the *center ranking* problem), aims to find a ranking (not necessarily from the input set) that minimizes the maximum distance to the input rankings. However, even for four input rankings, this problem is NP-hard (Dwork et al., WWW'01, and Biedl et al., Discrete Math.'09), and only a (folklore) polynomial-time 2-approximation algorithm is known for finding an optimal aggregate ranking under the commonly used *Kendall-tau distance* metric. Achieving a better approximation factor in polynomial time, ideally, a polynomial time approximation scheme (PTAS), is one of the major challenges.

This paper presents significant progress in solving this problem by considering the *Mallows model*, a classical probabilistic model. Our proposed algorithm outputs an $(1+\varepsilon)$-approximate aggregate ranking for any $\varepsilon > 0$, with high probability, as long as the input rankings come from a Mallows model, even in a streaming fashion. Furthermore, the same approximation guarantee is achieved even in the presence of *outliers*, presumably a more challenging task.

2012 ACM Subject Classification Theory of computation → Probabilistic computation; Theory of computation → Facility location and clustering; Theory of computation → Theory and algorithms for application domains

Keywords and phrases Rank Aggregation, Center Problem, Mallows Model, Approximation Algorithms, Clustering with Outliers

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.12

Funding This work received funding from an MoE AcRF Tier 2 grant (MOE-T2EP20221-0009) and Google South & South-East Asia Research Award.

1 Introduction

Ranking a set of candidates or a list of alternatives is ubiquitous in diverse fields ranging from social choice theory [7] to information retrieval [23]. Given a set of different, potentially conflicting rankings of a list of candidates, a fundamental task is to combine them to obtain a consensus ranking that best represents the preferences in the individual rankings. This process, known as *rank aggregation*, has applications in many areas, including voting systems, sports, databases, web search engines, collaborative filtering, machine translation, and statistics [19, 20, 41, 22]. One of the primary objectives is to mitigate apparent biases in individual rankings, such as search engine spam, strategic manipulation of web pages to achieve an undeserved high rank, or biases towards candidates/alternatives based on gender, ethnicity, and so on.

Rank aggregation is often formulated as an optimization problem. One popular version involves finding a ranking (not necessarily from the input set) that minimizes the maximum distance to the input rankings, known as the *maximum rank aggregation* problem in the

© Yan Hong Yao Alvin and Diptarka Chakraborty;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 12; pp. 12:1–12:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

literature. The optimization task considered here is the well-known *center* problem, with the underlying domain being the set of rankings/permutations. The general center problem dates back to the nineteenth century [44] and has been studied for various metric spaces, including Euclidean (both constant [36] and high dimension [4, 46]), Hamming [21, 30, 34], the edit metric [37], Jaccard distance [9], rankings [3, 5, 40, 13], and others. The hardness of the center problem varies with the metric space considered.

Several distance metrics have been explored to quantify the dissimilarity between two rankings or permutations. The *Kendall-tau distance* [27, 17, 26, 47, 48], which counts the number of inversions between two rankings (equivalently, the number of transpositions required by the *bubble sort* algorithm), is perhaps the most widely used metric. It is known to satisfy several desirable properties, such as neutrality, consistency, and the extended Condorcet property [26, 47]. In this paper, we focus on the maximum rank aggregation problem under the Kendall-tau metric, as is common in most of the rank aggregation literature [19, 1, 28, 3, 5]. Unfortunately, the problem is known to be NP-hard, even when there are only four input rankings [19, 5]. For most practical applications, computing an approximate maximum aggregate ranking suffices. A simple (folklore) algorithm that outputs one of the input rankings can achieve a 2-approximation factor, which is the case for any arbitrary metric space. However, finding a polynomial-time algorithm that can achieve a better than 2-approximation factor remains a significant open challenge. The ultimate goal is to develop a polynomial-time approximation scheme (PTAS) that can provide a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$.

Given the difficulty of the maximum rank aggregation problem, it is natural to consider this problem in the context of probabilistic models (i.e., when the input is generated from some probabilistic model). Various probabilistic ranking models have been proposed in the literature to capture preference or choice behavior, such as Bradley-Terry model [6], Luce model [33], Plackett-Luce model [39], and Cayley-Mallows model [16], along with several ranking algorithms and statistical analysis techniques [25, 43, 24]. Mallows model [35], a subclass of the Bradley-Terry model for noisy rankings, is a more tractable probabilistic ranking model. The model is defined by an underlying ranking π and a *dispersion* parameter $\phi \in (0,1)$, and the probability of observing a ranking σ decreases exponentially with respect to the Kendall-tau distance between π and σ and the dispersion parameter ϕ. Being a problem of utter importance, the rank aggregation problem under the Mallows model has already sought attention from the researchers [8, 10, 45], albeit mostly on minimizing the sum of distances (the *median* objective) instead of minimizing the maximum distance (that we consider in this paper), where the latter task is considered to be more challenging than the first one in the context of aggregating rankings.

1.1 Our Contribution

In this paper, we study the maximum rank aggregation problem when the input is sampled from the Mallows model. So far, nothing better than the folklore 2-approximation algorithm is known, even under this probabilistic ranking model. We make significant progress by providing an $(1+\varepsilon)$-approximation algorithm.

▶ **Theorem 1.** *Consider a $\phi \in (0,1)$, $\varepsilon > 0$ and a ranking π on n candidates. There is a polynomial-time algorithm that, given a set S of size at least $\frac{50}{(1-\phi)^2}\left(\log\frac{9}{\varepsilon\phi} + \log\frac{1}{1-\phi}\right)$ drawn from a Mallows model $M(\pi,\phi)$, outputs a permutation π_c that is a $(1+\varepsilon)$-approximate center (maximum aggregate) ranking with probability at least $2/3$.*

Although, for simplicity, we state the success probability to be 2/3, a slightly careful look into our analysis[1] reveals that we can actually attain a success probability of at least $1 - e^{-\Omega\left((1-\phi)^2|S|\right)}$. Our algorithm runs in $O(n^3 \log n)$ time, where the $O(\cdot)$ notation hides a constant that depends on $1/\varepsilon$ and $1/(1-\phi)$. The running time can further be reduced to $O(n \log^2 n)$ when the size of the input set is at least $\Omega(\log n)$. We also remark that as such, our algorithm does not need to know the parameter ϕ as long as the input set size satisfies the required minimum bound of $\frac{50}{(1-\phi)^2}\left(\log \frac{9}{\varepsilon\phi} + \log \frac{1}{1-\phi}\right)$. It is worth highlighting that when ϕ is close to 1, the Mallows model defines a distribution close to the uniform distribution. A set of rankings drawn from such a (close to) uniform distribution has little similarity; thus, aggregating them into a representative ranking has little importance. Therefore, the problem is more interesting when ϕ is away from 1, where our algorithm only needs a constant (dependent on parameters ε, ϕ) number of samples. We pose the problem of showing whether so many samples are essential to achieve a $(1+\varepsilon)$-approximation as an interesting open problem.

Our result consists of two parts. In the first part, we prove that when the input set is drawn from a Mallows model $M(\pi, \phi)$, the underlying ranking π gives an additive approximation of $\frac{n}{1-\phi} e^{-\frac{s(1-\phi)^2}{50}}$, where the input set is of size s (Theorem 5). Such an equivalence between the hidden ranking in the Mallows model and an approximate center ranking was not known before. We exploit the relation between the Mallows model and the *Repeated Insertion model* to show our result. This allows us to argue that the distances of rankings in the input set to π are almost the same (Lemma 7). To show the additive approximation guarantee, we consider an arbitrary optimal center ranking σ and then view it as being obtained from π by performing a sequence of swap operations of adjacent symbols (as in bubble sort). Ideally, one would like to show that such swaps would not reduce the (center) objective value by much. However, that is not true. We argue that such swaps can be categorized into two types. In the first one, on a pair of symbols involved in the swap, π and the majority of input rankings agree about their relative ordering. We argue that performing swaps on them cannot significantly improve the (center) objective value. Roughly speaking, the decrease in the distance caused by these swaps "almost equally" spreads out over all the input rankings. On the other hand, the second category constitutes all the pairs of symbols involved in swaps for which π and the majority of input rankings do not agree about their relative ordering. Although these swaps can improve the objective value, we argue that only a few such pairs exist (Lemma 9). Consequently, we derive that the objective value obtained by π is up to a small additive factor of the optimum objective value (i.e., the objective value obtained by σ).

Next, we show how to reconstruct a ranking π_c that is close (in the Kendall-tau distance) to π, without even knowing the parameter ϕ as long as there are sufficiently many input samples (Theorem 10). For that purpose, we build a *tournament graph* by looking into the pairwise relative ordering of the candidates in the majority of the input rankings and then use a PTAS by [28, 42] to solve the *feedback arc set* problem on this instance of tournament graph. Then, we remove the edge set returned by that PTAS from the constructed tournament graph and, finally, use the topological ordering of the resulting acyclic graph. The main difficulty lies in the analysis, where we show that this algorithm indeed reconstructs π up to some small distance. Interestingly, we only need a constant (depending on the dispersion parameter ϕ and the closeness requirement) number of input rankings for the reconstruction.

[1] The success probability of Lemma 9 can easily be improved to $1 - e^{-\Omega\left((1-\phi)^2|S|\right)}$ by adjusting certain parameters (and bounds) in the proof, and as a consequence, a similar success probability holds for both Theorem 5 and Theorem 10, and thus also for Theorem 1.

It is worth highlighting that the problem of approximate reconstruction of π (for the Mallows model) has been studied before [14]. However, our reconstruction result significantly differs from that of [14], primarily because [14] requires $\Omega(\log n)$ samples (with $\Omega(\cdot)$ hiding a factor that depends on $1/(1-\phi)$) while achieving a slightly different approximation guarantee, and thus the reconstruction result of [14] does not help us proving Theorem 1.

Since we only need constantly many inputs for the reconstruction, we can implement our algorithm in a streaming manner (when the input rankings arrive one by one in a streaming fashion) only using $O(n)$ space (note, the input size is $O(n|S|)$) by using a standard sampling technique. Furthermore, our algorithm can be extended to a more general setting (see Section 6) when the input set contains a small number of *outliers* chosen by the adversary (similar to that in [32]).

1.2 Related Works

In addition to the center objective that is the main focus of this paper, the rank aggregation problem has also been studied under the median objective function [19, 1]. This variant seeks to find a ranking (not necessarily from the input set) that minimizes the sum of distances to the input rankings. Under the Kendall-tau metric, the median ranking problem is also NP-hard [19], but it has a PTAS [28]. However, finding a similar PTAS or proving an impossibility result for the center variant remains a significant challenge. While the rank aggregation problem has been studied under Spearman's rho distance [19] or Ulam distance [11, 13, 12], Kendall-tau distance is the most prevalent for aggregating ranks.

The difficulty of the rank aggregation task has led researchers to explore data-driven approaches, such as using various probabilistic models [38, 8, 29]. The Mallows model is a classic probabilistic model for rankings, and [8] presented an algorithm that can find an optimal median of rankings drawn from a Mallows model in polynomial time. Other work has focused on estimating parameters and reconstructing the hidden permutation in more general mixtures of Mallows models [14, 2, 31, 15] or in the presence of outliers chosen by a malicious adversary [32]. [11] investigated a different probabilistic model for finding an approximate median ranking (although under the Ulam metric), but none of these results have yielded a better approximate center ranking than the folklore algorithm under the Mallows model.

2 Preliminaries

Let $[n]$ denote the set $\{1, 2, ..., n\}$. Let S_n denote the set of all rankings/permutations over $[n]$. We use the terms ranking and permutation interchangeably in this paper. Given a ranking π, let the notation $\pi(i)$ refer to the symbol at rank i in π. Let the notation $\pi^{-1}(a)$ refer to the rank of the symbol a in π. Given a pair of symbols a, b, we say $a \prec_\pi b$ if $\pi^{-1}(a) < \pi^{-1}(b)$. Throughout this paper, we will use the variables i, j when referring to ranks and a, b when referring to symbols. Also, note that all the logarithms in this paper are base e.

We consider the distance between the two rankings as their Kendall-tau distance.

▶ **Definition 2** (Kendall-tau Distance). *Given two rankings π, σ, the Kendall-tau distance, denoted by $d(\pi, \sigma)$, is the total number of ordered pairs $(a, b) \in [n] \times [n]$ such that $a \prec_\pi b$ and $b \prec_\sigma a$.*

Suppose we have some pair of symbols (a, b) and two rankings π and σ such that $a \prec_\pi b$ but $b \prec_\sigma a$. We say that the pair (a, b) is inverted in relative order in σ compared to π.

Center and c-approximate center

Given a set $S \subseteq S_n$ and ranking $\sigma \in S_n$, we refer to the center objective value of σ with respect to S as $\texttt{Obj}(S, \sigma) := \max_{\pi_r \in S} d(\sigma, \pi_r)$. Then an (optimal) center of a set $S \subseteq S_n$ is a ranking $\pi^* \in S_n$ that minimizes $\texttt{Obj}(S, \sigma)$, i.e., $\pi^* = \arg\min_{\sigma \in S_n} \texttt{Obj}(S, \sigma)$. We refer to $\texttt{Obj}(S, \pi^*)$ as $\mathsf{OPT}(S)$, or simply OPT when S is clear from the context. We call a ranking $\tilde{\pi}$ a *c-approximate center* (for some $c \geq 1$) for the set S iff $\texttt{Obj}(S, \tilde{\pi}) \leq c \cdot \mathsf{OPT}(S)$.

Mallows model

The Mallows model defines a distribution over rankings, where the probability of a ranking being sampled is dependent on its distance to a reference ranking.

▶ **Definition 3** (Mallows model). *A* Mallows model*, denoted by $M(\pi, \phi)$, on n elements is defined by two parameters, a ranking π and a dispersion parameter $\phi \in (0, 1)$.*

The probability that a ranking σ is sampled by the Mallows model is

$$\Pr[\sigma \mid \pi, \phi] = \frac{1}{Z(\pi, \phi)} \phi^{d(\pi, \sigma)},$$

where $Z(\pi, \phi) = \sum_{\rho \in S_n} \phi^{d(\pi, \rho)}$ is the normalization constant.

In the literature, sometimes the sampling probability is defined as $\frac{1}{Z(\pi,\phi)} \exp(-\phi \cdot d(\pi, \sigma))$ while allowing any $\phi > 0$. However, since both the definitions are equivalent, throughout this paper, we follow Definition 3. Observe, as $\phi \to 1$, the distribution induced by the Mallows model tends towards the uniform distribution over rankings. Also, the normalization constant $Z(\pi, \phi)$ is, in fact, independent of the ranking π (when considering the Kendall-tau distance, which is the case in this paper).

An alternative and mathematically equivalent way of formulating the Mallows model using the Kendall-tau distance is by using a more general *Repeated Insertion Model* [18]. This definition helps to establish many interesting properties of the sampling distribution.

▶ **Definition 4** (Repeated Insertion model). *Given a ranking π and a dispersion parameter $\phi \in (0, 1)$, the* Repeated Insertion model *creates a new ranking σ by the following iterative process: First, initialize σ to be empty. Then for each $i = 1, 2, \cdots, n$, take the element at $\pi(i)$ and insert it at rank j in σ with probability $\frac{\phi^{i-j}}{1+\phi+\cdots+\phi^{i-1}}$.*

3 Approximating a Center

In this section, we show that the hidden ranking π of the Mallows model is a "good" approximation to an (optimal) center.

▶ **Theorem 5.** *Consider a $\phi \in (0, 1)$ and a ranking $\pi \in S_n$. Let S be a set of rankings sampled from the Mallows model $M(\pi, \phi)$, of size s. Then*

$$\texttt{Obj}(S, \pi) \leq \mathsf{OPT}(S) + \frac{n}{1-\phi} e^{-\frac{s(1-\phi)^2}{50}} + o(n)$$

with probability at least $9/10 - 2s/n$.

We first provide a high-level overview of the proof. First, for all $\pi_r \in S$, $d(\pi_r, \pi)$ are the same up to some small factor (see Lemma 7). Next, let σ be an (arbitrary optimal) center of S. Now, we can view σ as being obtained from π by performing a sequence of swap

operations of adjacent symbols (as in bubble sort). Each such pair of symbols fall into two categories. For a particular such pair, either the majority of rankings in S agree with π on the pairwise (relative) order of that pair or the majority of rankings in S disagree with π on that pair. We argue that for all the pairs for which the majority of rankings agree, performing swaps on them cannot improve the objective value by much (roughly speaking, the decrease in the distance caused by these swaps "almost equally" spreads out over all the input rankings). On the other hand, if the majority of rankings disagree, we can improve the objective value by that particular swap; however, we show that there cannot be too many such pairs of symbols (see Lemma 9). Putting these together, we show that the objective value of σ cannot be much better than that of π.

Let us now start proving formally by first providing an upper and lower bound on the distance of any sampled ranking from π. For this section, fix

$$k = \frac{2}{\log(1/\phi)}.$$

Now, we bound the number of inversions created at each step of the Repeated Insertion Model.

▶ **Lemma 6.** *The probability that any step of the Repeated Insertion model creates at least $k \log n$ inversions compared to π is at most $\frac{1}{n}$.*

Proof. Recall that from the definition of the Repeated Insertion model (Definition 4), the probability that m inversions are created at the i-th step (for any $i \in [n]$) is equal to $\frac{\phi^m}{1+\phi+\ldots+\phi^{i-1}}$.

Therefore, the probability that at least $k \log n$ inversions are created is

$$\begin{aligned}\Pr[\text{At least } k\log n \text{ inversions are created}] &= \frac{\phi^{k\log n} + \phi^{k\log n+1} + \ldots + \phi^{i-1}}{1+\phi+\ldots+\phi^{i-1}} \\ &= \phi^{k\log n}\frac{1+\phi+\ldots+\phi^{i-1-k\log n}}{1+\phi+\ldots+\phi^{i-1}} \\ &\leq \phi^{k\log n} = \frac{1}{n^2}\end{aligned}$$

where the last equality follows since we set $k = \frac{2}{\log(1/\phi)}$. Taking a union bound over n steps gives the desired probability bound. ◀

Now, we prove a concentration bound on the distance of any sampled ranking from π, which we show by first bounding the expected number of inversions created by any step of the Repeated Insertion model.

▶ **Lemma 7.** *Consider a π_r drawn from $M(\pi, \phi)$. With probability at least $1 - \frac{2}{n}$,*

$$(1-\lambda)\frac{\phi}{1-\phi}n \leq d(\pi, \pi_r) \leq (1+\lambda)\frac{\phi}{1-\phi}n,$$

for some $\lambda = o(1)$.

Proof. We first consider the inversions caused by any step i of the Repeated Insertion model, where $j > k\log n$. Let the number of inversions at step j (where $j > k\log n$) in the model be represented by a random variable X_j. Let $X = \sum_{j=k\log n+1}^{n} X_j$. Let us now calculate $\mathbb{E}[X_j]$ for each $j > k\log n$.

By the definition of the Repeated Insertion model, the probability that at step j we have m inversions created is equal to $\frac{\phi^m}{1+\phi+...+\phi^{j-1}}$. Therefore the expected number of inversions caused by $\pi(j)$ with symbols $\pi(i)$ for all $i < j$ is equal to

$$\frac{1}{1+\phi+...+\phi^j}\sum_{i=0}^{j-1} i\phi^i = \frac{\phi L}{(1-\phi)L + j\phi^{j-1}}, \tag{1}$$

where $L = 1 + 2\phi + 3\phi^2 + ... + (j-1)\phi^{j-2}$.

Then by the linearity of expectation,

$$\mathbb{E}[X] = \sum_{j=k\log n+1}^{n} \mathbb{E}[X_j] = \left(\frac{\phi}{1-\phi} - \delta\right) n$$

where $\delta = O(1/n^2)$. The last equality follows since for $j > k\log n$ and $k = \frac{2}{\log(1/\phi)}$, $j\phi^{j-1} = O\left(\frac{1}{n^2}\right)$.

Let us now consider the good event $\mathcal{G}_1$ that none of the steps of the Repeated Insertion model creates more than $k\log n$ inversions. By Lemma 6, $\Pr[\bar{\mathcal{G}}_1] \leq 1/n$. From now on, we assume that the event $\mathcal{G}_1$ happens. Then the random variables X_j's take on values only in the range $[0, k\log n]$. Thus, by the standard Hoeffding bound,

$$\Pr\left[|X - \mathbb{E}[X]| \geq k\sqrt{n(\log n)^3}\right] \leq 2\exp\left(-2\log n\right).$$

That means given the event $\mathcal{G}_1$,

$$\Pr\left[X \in (1 \pm \lambda')\left(\frac{\phi}{1-\phi} - \delta\right) n\right] \geq 1 - \frac{2}{n^2}$$

where $\lambda' = o(1)$.

It is straightforward to see that the first $k\log n$ steps can create at most $k^2(\log n)^2$ inversions. Recall that $d(\pi, \pi_r)$ is equal to the total number of inversions created by all the steps of the Repeated Insertion Model while sampling π_r. Therefore,

$$(1-\lambda')\left(\frac{\phi}{1-\phi} - \delta\right) n \leq d(\pi, \pi_r) \leq (1+\lambda')\left(\frac{\phi}{1-\phi} - \delta\right) n + k^2(\log n)^2$$

with probability at least $1 - 2/n$ (since $\Pr[\bar{\mathcal{G}}_1] \leq 1/n$). Since $\delta = O(\frac{1}{n^2})$, the lemma follows. ◀

Next, we consider the set S of rankings sampled from the Mallows model $M(\pi, \phi)$. We show that there are not too many pairs of symbols (a, b) such that $a \prec_\pi b$, but the majority of rankings $\pi_r \in S$ have $b \prec_{\pi_r} a$. Let us start with a known result on the probability that a given pair is inverted in a sampled ranking compared to π.

▶ **Lemma 8** ([14]). *Let us consider a $j \in [1, n]$ and an $i < j$. The probability that a ranking π_r drawn from $M(\pi, \phi)$ has $\pi(j) \prec_{\pi_r} \pi(i)$ is equal to $\frac{\phi^k(\phi^{k+1} - (k+1)\phi + k)}{(1-\phi^{k+1})(1-\phi^k)}$, where $k = j - i$.*

For a set S drawn from $M(\pi, \phi)$, let us now consider the set P_S of pairs (a, b) that are inverted in relative order in at least $\frac{|S|}{2}$ rankings in S compared to π. More formally,

$$P_S := \{(a, b) \in [n] \times [n] \mid a \prec_\pi b \text{ but } b \prec_{\pi_r} a \text{ for at least } |S|/2 \text{ many } \pi_r \in S\}.$$

We show that the size of P_S would not be "too large" with a probability at least 9/10.

▶ **Lemma 9.** *Consider $\phi \in (0,1)$ and a set S drawn from $M(\pi,\phi)$, of size s. Then with a probability at least 9/10,*

$$|P_S| \leq \frac{n}{1-\phi} \exp\left(-\frac{s(1-\phi)^2}{50}\right).$$

Proof. Let H be a random variable whose value is equal to $|P_S|$.

First, we fix a specific value of $j \in [1,n]$. Let us consider an $i \in [0, j-1]$ and let $k = j - i$. Let I_k be a random variable whose value is equal to the number of rankings $\pi_r \in S$ such that $\pi(j) \prec_{\pi_r} \pi(i)$. Then it follows from Lemma 8 that

$$\begin{aligned}
\mu := \mathbb{E}\left[I_k\right] &= s \cdot \frac{\phi^k(\phi^{k+1} - (k+1)\phi + k)}{(1-\phi^{k+1})(1-\phi^k)} \\
&= s \cdot \phi^k \cdot \frac{k(1-\phi) - \phi(1-\phi^k)}{(1-\phi^{k+1})(1-\phi^k)} \\
&\leq s \cdot \phi^k \cdot \frac{k-\phi}{(1-\phi^{k+1})} \\
&\leq \frac{sk\phi^k}{1-\phi}.
\end{aligned}$$

Applying the standard Chernoff bound,

$$\begin{aligned}
\Pr\left[I_k \geq \frac{s}{2}\right] \leq 2\exp\left(\frac{-(\frac{s}{2\mu}-1)^2\mu}{3}\right) &\leq 2\exp\left(\frac{-(\frac{s}{3\mu})^2\mu}{3}\right) \\
&\leq 2\exp\left(-\frac{s^2}{27\mu}\right) \\
&\leq 2\exp\left(-\frac{s}{27}\cdot\frac{1-\phi}{k\phi^k}\right) \qquad \left(\text{as } \mu \leq \frac{sk\phi^k}{1-\phi}\right).
\end{aligned}$$

Let J_k be a random variable that takes on 1 if $I_k \geq \frac{s}{2}$; and 0 otherwise. Let $H_j = \sum_{k=1}^{j-1} J_k$. Then by the linearity of expectation,

$$\mathbb{E}[H_j] \leq \sum_{k=1}^{j-1} 2\exp\left(-\frac{s}{27}\cdot\frac{1-\phi}{k\phi^k}\right) = \sum_{k=1}^{j-1} 2\exp\left(-\frac{s(1-\phi)}{27}\right)^{\frac{1}{k}\phi^{-k}}. \tag{2}$$

Let $\theta := 1-\phi$. We have that for any k,

$$\frac{1}{k}\phi^{-k} = \frac{1}{k(1-\theta)^k} \geq \frac{1}{k}(1+k\theta) \geq \theta. \tag{3}$$

Let $K = \frac{4}{\theta}\log(4/\theta)$. Then, it is not hard to see that

$$\text{for all } k \geq K, \quad \frac{1}{k}\phi^{-k} \geq k. \tag{4}$$

For brevity, let $t := \exp\left(-\frac{s(1-\phi)}{27}\right)$. Then we rewrite Equation 2 as follows

$$
\begin{aligned}
\mathbb{E}[H_j] &\leq \sum_{k=1}^{K-1} 2t^{\frac{1}{k}\phi^{-k}} + \sum_{k=K}^{j-1} 2t^{\frac{1}{k}\phi^{-k}} \\
&\leq \sum_{k=1}^{K-1} 2t^{\theta} + \sum_{k=K}^{j-1} 2t^{k} && \text{(by Equation 3 and Equation 4)} \\
&\leq 2Kt^{\theta} + 2t^{K}\frac{1}{1-t} \\
&\leq 2K\exp\left(-\frac{\theta^2 s}{27}\right) + 2\exp\left(-\frac{\theta s K}{27}\right)\frac{1}{1-\exp\left(-\frac{s\theta}{27}\right)} \\
&\leq \frac{12}{\theta}\log\frac{4}{\theta}\exp\left(-\frac{\theta^2 s}{27}\right) && \left(\text{by setting } K = \frac{4}{\theta}\log(4/\theta)\right) \\
&\leq \frac{1}{\theta}\exp\left(-\frac{\theta^2 s}{40}\right).
\end{aligned}
$$

Since $H = \sum_j H_j$, by the linearity of expectation, we have

$$\mathbb{E}[H] \leq \frac{n}{\theta}\exp\left(-\theta^2 s/40\right).$$

Then by Markov's inequality,

$$\Pr\left[H \geq \frac{10n}{\theta}\exp\left(-\theta^2 s/40\right)\right] \leq 1/10$$

and the lemma follows. ◀

We are now ready to prove Theorem 5.

Proof of Theorem 5. Let σ be an (arbitrary optimal) center of the set S (where $|S| = s$). Let us consider a sequence of swaps (or inversion) of adjacent pair(s) of symbols to convert the ranking π into the ranking σ. (For example, while performing bubble sort on π with σ as the total order of the symbols, we get a sequence of swaps of consecutive symbols.) Further, let this sequence of swaps converts $\pi \to \pi^1 \to \pi^2 \to \cdots \to \sigma$. Next, we categorize each such pair of symbols into two cases.

The first case is when the pair of symbols (a, b) is inverted in relative order in at most half the rankings of S compared to π (i.e., $(a, b) \notin P_S$). Suppose the swap of adjacent pair (a, b) is performed in the intermediate step while converting $\pi^j \to \pi^{j+1}$ (for some j). Observe, for at most $s/2$ rankings $\pi_r \in S$, $d(\pi^{j+1}, \pi_r) = d(\pi^j, \pi_r) - 1$, whereas for all other rankings $\pi_f \in S$, $d(\pi^{j+1}, \pi_f) = d(\pi^j, \pi_f) + 1$. Hence

$$\sum_{\pi_i \in S} d(\pi^{j+1}, \pi_i) \geq \sum_{\pi_i \in S} d(\pi^j, \pi_i). \tag{5}$$

The second case is when the pair of symbols (a, b) is inverted in relative order in more than half the rankings of S compared to π (i.e., $(a, b) \in P_S$). Again, suppose the swap of adjacent pair (a, b) is performed in the intermediate step while converting $\pi^j \to \pi^{j+1}$ (for some j). Since in the worst case, all the rankings in S have a distance of one less to π^{j+1} than to π^j,

$$\sum_{\pi_i \in S} d(\pi^{j+1}, \pi_i) \geq \sum_{\pi_i \in S} d(\pi^j, \pi_i) - s. \tag{6}$$

Let us now assume that the events of Lemma 7 and Lemma 9 occur. More specifically, let us consider the good event $\mathcal{G}$ that

- For all $\pi_r \in S$, $(1-\lambda)\frac{\phi}{1-\phi}n \le d(\pi, \pi_r) \le (1+\lambda)\frac{\phi}{1-\phi}n$, for some $\lambda = o(1)$, and
- $|P_S| \le \frac{n}{1-\phi} \exp\left(-\frac{s(1-\phi)^2}{50}\right)$.

By a union bound, $\Pr[\bar{\mathcal{G}}] \le 1/10 + 2s/n$. From now on, let us assume that $\mathcal{G}$ occurs.

In the process to transforming $\pi \to \pi^1 \to \pi^2 \to \cdots \to \sigma$, by repeatedly applying Equation 5 and Equation 6, we get that

$$d(\pi_i, \sigma) \ge \left(\sum_{\pi_i \in S} d(\pi_i, \pi) - \frac{sn}{1-\phi} e^{\frac{-s(1-\phi)^2}{50}} \right). \tag{7}$$

Hence, we derive that

$$\begin{aligned}
\mathsf{OPT}(S) = \mathtt{Obj}(S, \sigma) &\ge \frac{1}{s} \sum_{\pi_i \in S} d(\pi_i, \sigma) && \text{(by averaging)} \\
&\ge \frac{1}{s} \left(\sum_{\pi_i \in S} d(\pi_i, \pi) - \frac{sn}{1-\phi} e^{\frac{-s(1-\phi)^2}{50}} \right) && \text{(by Equation 7)} \\
&\ge \frac{1}{s} \sum_{\pi_i \in S} d(\pi_i, \pi) - \frac{n}{1-\phi} e^{\frac{-s(1-\phi)^2}{50}} \\
&\ge \min_{\pi_i \in S} d(\pi_i, S) - \frac{n}{1-\phi} e^{\frac{-s(1-\phi)^2}{50}} \\
&\ge (1-\lambda)\frac{\phi}{1-\phi} n - \frac{n}{1-\phi} e^{\frac{-s(1-\phi)^2}{50}} && \text{(assuming the event } \mathcal{G}\text{)}.
\end{aligned}$$

Recall that assuming the event $\mathcal{G}$, we have that $\mathtt{Obj}(S, \pi) \le (1+\lambda)\frac{\phi}{1-\phi}n$. Therefore,

$$\begin{aligned}
\mathtt{Obj}(S, \pi) &\le \mathsf{OPT}(S) + 2\lambda \frac{\phi}{1-\phi} n + \frac{n}{1-\phi} e^{\frac{-s(1-\phi)^2}{50}} \\
&\le \mathsf{OPT}(S) + \frac{n}{1-\phi} e^{\frac{-s(1-\phi)^2}{50}} + o(n) && \text{(as } \lambda = o(1)\text{)}
\end{aligned}$$

which concludes the proof. ◀

4 Reconstructing the Hidden Ranking

In this section, we discuss an algorithm that, given samples from a Mallows model, approximately reconstructs the hidden ranking/permutation π in polynomial time.

▶ **Theorem 10.** *Consider $\phi \in (0,1)$, $\alpha > 0$ and a $\pi \in S_n$. There exists a polynomial-time algorithm that, given a set S of size $s \ge \frac{50}{(1-\phi)^2}\left(\log \frac{3}{\alpha} + \log \frac{1}{1-\phi}\right)$ drawn from $M(\pi, \phi)$, computes a ranking π_c such that $d(\pi_c, \pi) \le \alpha n$ with probability at least 9/10.*

It is important to note that the number of samples required in the above theorem is only a constant (that depends on the dispersion parameter ϕ and the closeness parameter α). It is worth mentioning that if the size of the input set is $\Omega(\log n)$, then it is easy to recover the hidden ranking exactly. We now prove Theorem 10.

Description of the algorithm

We start by describing the algorithm. We first build a graph G on the vertex set $[n]$. Add a directed edge from vertex a to vertex b if at least half the rankings in S rank symbol a before b, i.e., for at least $s/2$ samples $\pi_r \in S$, $a \prec_{\pi_r} b$. Observe the graph obtained is a *tournament*

Algorithm 1 ApproxReconstruct.

Input: $S \subset S_n$ of size s
Output: A ranking π_c over $[n]$

$G \leftarrow ([n], E)$ where $E = \{(a, b) \mid a$ is ranked before b in at least $\frac{s}{2}$ rankings$\}$.
Run the PTAS of [28] to get a subset F of edges.
G' is obtained by removing the edge set F from G.
$\pi_c \leftarrow$ topological ordering of G'.
return π_c.

graph[2]. However, it may contain a cycle. Next, we want to remove as few edges as possible from G to make it acyclic, which is the well-known *feedback arc set* problem. Kenyon-Mathieu and Schudy [28, 42] designed a PTAS for the feedback arc set problem over tournament graphs. We run that PTAS to obtain a $(1+\varepsilon)$-approximation of the optimal feedback arc set and remove those edges to form a directed acyclic graph G'. Finally, we obtain a topological order on the vertices of G' and return this as ranking π_c. (See Algorithm 1.)

Approximation guarantee on the reconstructed ranking

We call an edge (b, a) in G *bad* if $a \prec_\pi b$. Let B be the set of all the bad edges. Recall, we define P_S to be the set of pairs (a, b) that are inverted in relative order in at least $\frac{|S|}{2}$ rankings in S compared to π. By the construction of the graph G, the pairs of symbols in P_S essentially constitute the bad edge set B. Thus $|B| = |P_S|$.

So by Lemma 9 we have that the size of B is at most $\frac{n}{1-\phi} \exp\left(-\frac{s(1-\phi)^2}{50}\right)$ with probability at least 9/10. For $s \geq \frac{50}{(1-\phi)^2}\left(\log\frac{3}{\alpha} + \log\frac{1}{1-\phi}\right)$, we have that

$$\begin{aligned} |B| &\leq \frac{n}{1-\phi} \exp\left(-\log\frac{3}{\alpha} - \log\frac{1}{1-\phi}\right) \\ &\leq \frac{\alpha}{3} n. \end{aligned} \tag{8}$$

It is straightforward to observe that removing the set B of bad edges from G makes it acyclic. So $|B|$ (which is at most $\alpha n/3$ by Equation 8) serves as an upper bound on the size of an optimal feedback arc set of G. Since by applying the PTAS of [28], we get the subset F of edges to remove from G to make it acyclic, $|F| \leq (1+\varepsilon)\alpha n/3$, for any $\varepsilon > 0$.

Since π_c is the resulting ranking from taking a topological ordering of the resulting directed acyclic graph $G' = G - F$,

$$d(\pi_c, \pi) \leq |B| + |F| \leq (2+\varepsilon) \cdot \frac{\alpha n}{3}$$

which completes the proof of Theorem 10.

▶ Remark 11. Note the running time of constructing the graph and finding a topological ordering is $O(sn^2)$. However, since for the reconstruction, $s = \mathsf{poly}(\log(1/\alpha), 1/(1-\phi))$ samples suffice, the running time is $O(n^2)$, where $O(\cdot)$ hides the constant term that depends (polynomially) on $1/\alpha$ and $1/(1-\phi)$. This is dominated by the running time of the PTAS for the feedback arc set on tournaments from [28, 42], which is $O\left(\frac{n^3 \log n}{\varepsilon}\right) + n2^{O(\varepsilon^{-6})}$, where ε comes from the $(1+\varepsilon)$-factor of PTAS. We can significantly improve the running time of

[2] A directed graph is called a *tournament* if there is a (directed) edge between every pair of vertices.

this algorithm by increasing the input size required by a constant factor, and then instead of running the PTAS algorithm, we can run a randomized algorithm by [1] that finds a 3-approximation of the optimal feedback arc set and runs in only $O(n \log n)$ time.

Exact reconstruction of the hidden ranking

In Theorem 10, we provide an approximate reconstruction algorithm using only constantly many samples. It is worth mentioning that a polynomial time exact reconstruction algorithm is already known due to [14]; however, that requires $\Omega(\log n)$ samples. We also want to mention that as long as the sampled set is of size at least $\frac{12(1+\phi)^2}{(1-\phi)^2} \log n$, for every pair of symbols (a, b), their relative ordering in at least half of the samples will be exactly the same as that in π with high probability (with probability at least $1 - 1/\mathsf{poly}(n)$). Hence, by using any standard comparison-based sorting algorithm, we can reconstruct π exactly in time $O(n \log^2 n)$. Although this simple approach is quite standard and well-known, just for the sake of completeness, we provide detailed proof of this simple argument in Appendix B.

5 Completing the proof of Theorem 1

We first show a lower bound of $\mathsf{OPT}(S)$ for any set S drawn from a Mallows model.

▶ **Lemma 12.** *Consider a set S (of size at least two) drawn from $M(\pi, \phi)$. Then, with probability at least $1 - \frac{6}{n}$, $\mathsf{OPT}(S) \geq \phi n - o(n)$.*

Proof. Let $\pi_1, \pi_2 \in S$. Let I be the set of pairs that are inverted in relative order in both π_1 and π_2 compared to π.

First, take some fixed $j \in [1, n]$ and consider the step j in the Repeated Insertion model. Let B_j be a random variable that takes on a value equal to the size of the set

$$\{i \mid i < j, \pi(j) \prec_{\pi_1} \pi(i), \pi(j) \prec_{\pi_2} \pi(i)\}.$$

Let C_j be a random variable that takes on a value t if

- $\pi(j)$ is placed at rank $j - t$ in π_1 (i.e., creating t inversions), and
- $\pi(j)$ is placed at some rank $j - r$ for $t \leq r \leq j - 1$ in π_2 (i.e., creating at least t inversions).

Observe that if $|\{i \mid i < j, \pi(j) \prec_{\pi_1} \pi(i), \pi(j) \prec_{\pi_2} \pi(i)\}| = t$, then the insertion of the symbol at rank j must have created at least t inversions in both π_1 and π_2. Therefore for any $t > 0$, $\Pr[B_j \geq t] \leq \Pr[C_j \geq t]$.

Now, let $B = \sum_{j=1}^{n} B_j$ and $C = \sum_{j=1}^{n} C_j$. Observe that the size of the set I is equal to B. Recall that each step of the Repeated Insertion model is independent of each other.

Consider $\Pr[C_j = m]$ for some $j \in [1, n]$ and $m \in [0, j - 1]$. As sampling both rankings is independent, by Definition 4, we get that

$$\Pr[C_j = m] = \frac{\phi^m}{1 + \phi + ... + \phi^{j-1}} \cdot \frac{\phi^m + ... + \phi^{j-1}}{1 + \phi + ... + \phi^{j-1}}.$$

Thus, we deduce that

$$\begin{aligned}\mathbb{E}[C_j] &= \sum_{m=1}^{j-1} m \cdot \frac{\phi^m}{1 + \phi + ... + \phi^{j-1}} \cdot \frac{\phi^m + ... + \phi^{j-1}}{1 + \phi + ... + \phi^{j-1}} \\ &\leq \sum_{m=1}^{j-1} m \cdot \phi^m \frac{\phi^m}{1 + \phi + ... + \phi^{j-1}}\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{\phi}{1+\phi+\ldots+\phi^{j-1}} \sum_{m=1}^{j-1} m\phi^m \\
&\leq \phi \cdot \frac{\phi + 2\phi^2 \ldots + (j-1)\phi^{j-1}}{1+\phi+\ldots+\phi^{j-1}} \\
&\leq \phi \cdot \frac{\phi(1+2\phi+\ldots+(j-1)\phi^{j-2})}{(1-\phi)(1+2\phi+\ldots+(j-1)\phi^{j-2}) + i\phi^{j-1}} \\
&\leq \frac{\phi^2}{1-\phi}.
\end{aligned}
$$

By the linearity of expectation, $\mathbb{E}[C] \leq \frac{\phi^2}{1-\phi} n$.

Let us now assume that the events of Lemma 6 and Lemma 7 occur. More specifically, let us consider the good event $\mathcal{G}_2$ that

- None of the steps of the Repeated Insertion model creates more than $k \log n$ inversions, and
- For $r = 1, 2$, $(1-\lambda)\frac{\phi}{1-\phi} n \leq d(\pi, \pi_r) \leq (1+\lambda)\frac{\phi}{1-\phi} n$, for some $\lambda = o(1)$.

By a union bound of the failure probability of Lemma 6 and Lemma 7, $\Pr[\bar{\mathcal{G}_2}] \leq 5/n$. From now on, let us assume that $\mathcal{G}_2$ occurs.

So the random variables C_j's take on values in the range $[0, k \log n]$. Thus, by applying the standard Hoeffding bound,

$$\Pr\left[C - \mathbb{E}[C] \geq k\sqrt{n}(\log n)^{3/2}\right] \leq \exp(-2\log n). \tag{9}$$

So we have that given the event $\mathcal{G}_2$ occurs, with a probability at least $1 - 1/n^2$,

$$|I| \leq (1+\gamma)\frac{\phi^2}{1-\phi} n, \text{ for some } \gamma = o(1). \tag{10}$$

Let $\pi^*_{1,2}$ be an (arbitrary optimal) center of π_1 and π_2. Then by the triangle inequality,

$$d(\pi_1, \pi^*_{1,2}) \geq \frac{1}{2} d(\pi_1, \pi_2).$$

Also, by definition, $d(\pi_1, \pi_2) = d(\pi_1, \pi) + d(\pi_2, \pi) - 2|I|$. Further, observe that $\mathsf{OPT}(S) \geq d(\pi_1, \pi^*_{1,2})$ (as each of the other rankings in S can only increase the objective value).

Therefore,

$$
\begin{aligned}
\mathsf{OPT}(S) &\geq \frac{d(\pi_1, \pi) + d(\pi_2, \pi) - 2|I|}{2} \\
&\geq (1-\lambda)\frac{\phi}{1-\phi} n - |I| && \text{(assuming the event } \mathcal{G}_2) \\
&\geq (1-\lambda)\frac{\phi}{1-\phi} n - (1+\gamma)\frac{\phi^2}{(1-\phi)} n && \text{(by Equation 10)} \\
&\geq \frac{\phi}{1-\phi} n(1 - \lambda - (1+\gamma)\phi) && (11) \\
&\geq \phi n - o(n) && \text{(as } \lambda \text{ and } \gamma \text{ are } o(1)). \quad (12)
\end{aligned}
$$

The success probability follows from the failure probability of Equation 10 and $\Pr[\bar{\mathcal{G}_2}] \leq 5/n$. ◀

Proof of Theorem 1. Set $\alpha = \frac{\varepsilon\phi}{3}$. Recall that by Theorem 10, we find a permutation π_c such that $d(\pi_c, \pi) \leq \alpha n$. Next, recall that from Theorem 5 we have that given s samples,

$$\begin{aligned}\mathtt{Obj}(S, \pi) &\leq \mathsf{OPT}(S) + \frac{n}{1-\phi} e^{-\frac{s(1-\phi)^2}{50}} + o(n) \\ &\leq \mathsf{OPT}(S) + \frac{\alpha}{3} n + o(n) \end{aligned} \tag{13}$$

where the last inequality follows for $s \geq \frac{50}{(1-\phi)^2}\left(\log \frac{3}{\alpha} + \log \frac{1}{1-\phi}\right)$.

So, finally, we get that

$$\begin{aligned}\mathtt{Obj}(S, \pi_c) &\leq \mathtt{Obj}(S, \pi) + d(\pi, \pi_c) && \text{(by the triangle inequality)} \\ &\leq \mathsf{OPT}(S) + \frac{4}{3}\alpha n + o(n) && \text{(by Equation 13)} \\ &\leq \mathsf{OPT}(S) + \frac{\varepsilon\phi}{2} n && \text{(for } \alpha = \varepsilon\phi/3) \\ &\leq \mathsf{OPT}(S) + \varepsilon \cdot \mathsf{OPT}(S) && \text{(by Lemma 12)} \\ &\leq (1+\varepsilon)\mathsf{OPT}(S). \end{aligned}$$

Since the failure probabilities of Theorem 5, Theorem 10 and Lemma 12 are $1/10 + 2s/n$, $1/10$ and $6/n$ respectively, by a union bound, we have that all three simultaneously hold with probability at least $2/3$, for large enough n (since we can assume that $s = \frac{50}{(1-\phi)^2}\left(\log \frac{3}{\alpha} + \log \frac{1}{1-\phi}\right)$).

It follows from the discussion in Remark 11 that the running time of our algorithm is $O(n^3 \log n)$, where the $O(\cdot)$ notation hides a constant that depends on $1/\varepsilon$ and $1/(1-\phi)$. The running time can further be reduced to $O(n \log^2 n)$ when the size of the input set is at least $\Omega(\log n)$. ◀

6 Extension to Input with Outliers

Our algorithm can be extended to the scenario when a small number of *outliers* is also present in the input set. In the context of the clustering (or center) problem, outliers are the data points that are not part of the intended cluster. In the case of the probabilistic generating model, outliers can be considered a set of adversarially chosen points (that do not come from the generating model). Formally, we consider the following *Strong Contamination Model*. In this model, a set of s samples are first generated from a Mallows model $M(\pi, \phi)$ (for some $\pi \in S_n$ and $\phi \in (0,1)$). Then, an arbitrary subset of size at most δs (for some $\delta \in [0,1)$) is replaced with an arbitrary set of rankings in S_n. We call this resulting set (π, ϕ, δ)-*corrupted sample*. This model was previously considered in the literature (e.g., see [32]).

Given a set S of rankings, the δ-*maximum rank aggregation with outliers* problem seeks to find a ranking that minimizes the maximum distance to a subset $S' \subseteq S$ of size at least $(1-\delta)|S|$. We show that we can find a $(1+\varepsilon)$-approximate δ -maximum rank aggregation with outliers for (π, ϕ, δ)-corrupted sample sets.

▶ **Theorem 13.** *Consider a $\phi \in (0,1)$, $\varepsilon > 0$, $\delta \in (0, 0.5)$ and a ranking π on n candidates. There is a polynomial-time algorithm that, given a (π, ϕ, δ)-corrupted sample set S of size at least* $\mathsf{poly}\,(1/\varepsilon, 1/(1-\phi), 1/(0.5-\delta)) \log n$*, outputs a ranking π_c that is a $(1+\varepsilon)$-approximation to the δ-maximum rank aggregation with outliers problem with probability at least $2/3$.*

It is worth noting that the running time mentioned in the above theorem is $n^{O(1)}$ where $O(\cdot)$ notation hides a constant that depends (polynomially) on $\log(1/\varepsilon)$, $1/(1-\phi)$ and $1/(0.5-\delta)$. The brief idea for this extension is to first sample $O(\log n)$ rankings from the input uniformly at random. Then, we try out all possible subsets of this sample set and apply the algorithm from Theorem 10 on each of these subsets. We treat all of these computed rankings as potential centers. Finally, output the best one (that minimizes the objective value) among all these potential centers. We argue that the output will be a $(1+\varepsilon)$-approximation of the optimal solution. The key (though simple) observation is that it suffices to use only $O(\log n)$ samples in Theorem 1, and also, in polynomial time, we can try out all possible subsets. We defer the details to Appendix A.

7 Conclusion

This paper focuses on the maximum rank aggregation, also known as the center problem, concerning the Kendall-tau metric on inputs generated from the well-established Mallows model. Although a folklore algorithm can compute a 2-approximate center in any metric space, no better approximation guarantee is known for the Kendall-tau metric despite its significant importance in various applications. Therefore, finding an $(1+\varepsilon)$-approximation algorithm for this problem remains an essential open problem. This paper proposes a data-driven algorithm that provides a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$, given that the input set is sampled from a Mallows model with dispersion parameter $\phi \in (0,1)$, and the number of input rankings is at least $\mathsf{poly}(\log(1/\varepsilon), 1/(1-\phi))$. The proposed algorithm can also be extended to datasets containing a few adversarially chosen outliers.

An apparent open direction is to obtain similar results without any assumptions about the input set. Even breaking below the 2-factor in the worst-case would be an exciting direction. Another interesting open problem is to show whether the number of samples required to achieve $(1+\varepsilon)$-approximation in Theorem 1 is tight. Also, developing a similar result for other probabilistic generating models for rankings, as in [29], would be interesting.

References

1. Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):1–27, 2008.
2. Pranjal Awasthi, Avrim Blum, Or Sheffet, and Aravindan Vijayaraghavan. Learning mixtures of ranking models. *Advances in Neural Information Processing Systems*, 27, 2014.
3. Christian Bachmaier, Franz J. Brandenburg, Andreas Gleißner, and Andreas Hofmeier. On the hardness of maximum rank aggregation problems. *Journal of Discrete Algorithms*, 31:2–13, 2015. 24th International Workshop on Combinatorial Algorithms (IWOCA 2013).
4. Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*, pages 250–257, 2002.
5. Therese Biedl, Franz J Brandenburg, and Xiaotie Deng. On the complexity of crossings in permutations. *Discrete Mathematics*, 309(7):1813–1823, 2009.
6. Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
7. Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.
8. Mark Braverman and Elchanan Mossel. Sorting from noisy information. *arXiv preprint*, 2009. `arXiv:0910.1191`.

9 Marc Bury and Chris Schwiegelshohn. On finding the jaccard center. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017.

10 Ioannis Caragiannis, Ariel D Procaccia, and Nisarg Shah. When do noisy votes reveal the truth? *ACM Transactions on Economics and Computation (TEAC)*, 4(3):1–30, 2016.

11 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Approximating the median under the ulam metric. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 761–775. SIAM, 2021.

12 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Clustering permutations: New techniques with streaming applications. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 31:1–31:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

13 Diptarka Chakraborty, Kshitij Gajjar, and Agastya Vibhuti Jha. Approximating the center ranking under ulam. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

14 Flavio Chierichetti, Anirban Dasgupta, Ravi Kumar, and Silvio Lattanzi. On reconstructing a hidden permutation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.604.

15 Fabien Collas and Ekhine Irurozki. Concentric mixtures of mallows models for top-k rankings: sampling and identifiability. In *International Conference on Machine Learning*, pages 2079–2088. PMLR, 2021.

16 Anindya De, Ryan O'Donnell, and Rocco Servedio. Learning sparse mixtures of rankings from noisy information. *arXiv preprint*, 2018. arXiv:1811.01216.

17 Persi Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(2):262–268, 1977. URL: http://www.jstor.org/stable/2984804.

18 Jean-Paul Doignon, Aleksandar Pekeč, and Michel Regenwetter. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika*, 69(1):33–54, 2004.

19 Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10*, pages 613–622, 2001. doi:10.1145/371920.372165.

20 Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, pages 301–312, 2003.

21 Moti Frances and Ami Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.

22 David F Gleich and Lek-heng Lim. Rank aggregation via nuclear norm minimization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 60–68, 2011.

23 Donna Harman. Ranking algorithms. In William B. Frakes and Ricardo A. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 363–392. Prentice-Hall, 1992.

24 Kenneth Hung and William Fithian. Rank verification for exponential families. *The Annals of Statistics*, 47(2):758–782, 2019.

25 David R Hunter. Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406, 2004.

26 John G Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

27 Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

28 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pages 95–103, 2007.

29 Ashish Khetan and Sewoong Oh. Data-driven rank breaking for efficient rank aggregation. In *International Conference on Machine Learning*, pages 89–98. PMLR, 2016.

30 Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *Journal of the ACM (JACM)*, 49(2):157–171, March 2002.

31 Allen Liu and Ankur Moitra. Efficiently learning mixtures of mallows models. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 627–638. IEEE, 2018.

32 Allen Liu and Ankur Moitra. Robust voting rules from algorithmic robust statistics. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3471–3512. SIAM, 2023.

33 R Duncan Luce. Individual choice behavior, 1959.

34 Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. *SIAM Journal on Computing*, 39(4):1432–1443, 2010.

35 Colin L Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

36 Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM (JACM)*, 31(1):114–127, 1984.

37 François Nicolas and Eric Rivals. Complexities of the centre and median string problems. In *Combinatorial Pattern Matching, 14th Annual Symposium, CPM 2003, Morelia, Michocán, Mexico, June 25-27, 2003, Proceedings*, pages 315–327, 2003.

38 Vasyl Pihur, Susmita Datta, and Somnath Datta. Weighted rank aggregation of cluster validation measures: a monte carlo cross-entropy approach. *Bioinformatics*, 23(13):1607–1615, 2007.

39 Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2):193–202, 1975.

40 V Yu Popov. Multiple genome rearrangement by swaps and by element duplications. *Theoretical computer science*, 385(1-3):115–126, 2007.

41 Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, 2007.

42 Warren Schudy. *Approximation schemes for inferring rankings and clusterings from pairwise data*. Ph.D. diss., Brown University, 2012.

43 Nihar B Shah and Martin J Wainwright. Simple, robust and optimal ranking from pairwise comparisons. *The Journal of Machine Learning Research*, 18(1):7246–7283, 2017.

44 James Joseph Sylvester. A question in the geometry of situation. *Quarterly Journal of Pure and Applied Mathematics*, 1(1):79–80, 1857.

45 Wenpin Tang. Mallows ranking models: maximum likelihood estimate and regeneration. In *International Conference on Machine Learning*, pages 6125–6134. PMLR, 2019.

46 E Alper Yildirim. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization*, 19(3):1368–1391, 2008.

47 H Peyton Young. Condorcet's theory of voting. *American Political science review*, 82(4):1231–1244, 1988.

48 H Peyton Young and Arthur Levenglick. A consistent extension of condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300, 1978.

A Detailed Description of Extension to Input with Outliers

We start with formally defining the input (sample) generating model and the problem statement.

▶ **Definition 14** (Strong Contamination Model). *Consider any $\phi, \delta \in (0,1)$ and a ranking $\pi \in S_n$. A set of samples S of size s is said to be a (π, ϕ, δ)-*corrupted sample set *if it is generated as follows: First, sample s rankings from $M(\pi, \phi)$. Then, a malicious adversary observes these s rankings and replaces up to δs of them with any arbitrary rankings it chooses, and then returns the resulting set of s rankings in an arbitrary order.*

Now, let us formally define the δ-maximum rank aggregation with outliers problem. Given a set S of rankings, the *δ-maximum rank aggregation with outliers* problem seeks to find a ranking (not necessarily from S) that minimizes the maximum distance to a subset $S' \subseteq S$ of size at least $(1-\delta)|S|$.

Given a ranking ρ and set S, we define the objective value of ρ on S for the δ-maximum rank aggregation with outliers problem as

$$\mathtt{Obj}_\delta(S, \rho) := \min_{S' \subseteq S : |S'| \geq (1-\delta)|S|} \max_{\pi_r \in S'} d(\pi_r, \rho).$$

Recall, $\mathtt{Obj}(S', \rho) = \max_{\pi_r \in S'} d(\pi_r, \rho)$. So we can simply say

$$\mathtt{Obj}_\delta(S, \rho) = \min_{S' \subseteq S : |S'| \geq (1-\delta)|S|} \mathtt{Obj}(S', \rho).$$

Let us denote the optimum objective value for the δ-maximum rank aggregation with outliers problem for a set S by $\mathsf{OPT}_\delta(S)$. We call a ranking $\tilde{\pi}$ a *c-approximate δ-maximum aggregate ranking with outliers* (for some $c \geq 1$) for the set S iff $\mathtt{Obj}_\delta(S, \tilde{\pi}) \leq c \cdot \mathsf{OPT}_\delta(S)$, i.e., there exists a subset $T \subseteq S$ with $|T| \geq (1-\delta)|S|$ such that $\mathtt{Obj}(T, \tilde{\pi}) \leq c \cdot \mathsf{OPT}_\delta(S)$. Note when $\delta = 0$, the definition coincides with that without outliers. More specifically, we have $\mathtt{Obj}_0(S, \rho) = \mathtt{Obj}(S, \rho)$ and $\mathsf{OPT}_0(S) = \mathsf{OPT}(S)$.

▶ **Theorem 13.** *Consider a $\phi \in (0,1)$, $\varepsilon > 0$, $\delta \in (0, 0.5)$ and a ranking π on n candidates. There is a polynomial-time algorithm that, given a (π, ϕ, δ)-corrupted sample set S of size at least $\mathsf{poly}\left(1/\varepsilon, 1/(1-\phi), 1/(0.5-\delta)\right) \log n$, outputs a ranking π_c that is a $(1+\varepsilon)$-approximation to the δ-maximum rank aggregation with outliers problem with probability at least $2/3$.*

Before proving the above theorem, let us introduce a few notations and show a few lemmas that will help us in proving the above theorem. Let $G \subseteq S$ be the set of rankings sampled from the Mallows model $M(\pi, \phi)$.

▶ **Lemma 15.** *Consider a $\phi \in (0,1)$, $\alpha > 0$, $\delta \in (0, 0.5)$, a ranking $\pi \in S_n$. There is a polynomial-time algorithm that, given a (π, ϕ, δ)-corrupted sample set S of size at least $\mathsf{poly}(\log(1/\alpha), 1/(1-\phi), 1/(0.5-\delta)\}) \log n$, constructs a set of rankings P of size $\mathsf{poly}(n)$ such that there exists $\pi_c \in P$ satisfying $d(\pi_c, \pi) \leq \alpha n$ with probability at least 9/10.*

Proof. Let $s = |S|$. Set $p = \frac{2s' \log n}{s}$ where s' is the minimum required size of the input set in Theorem 10, multiplied with the factor $1/(0.5-\delta)^2$.

Now, sample each ranking in S independently uniformly at random with probability p, and let this sample set be T. It is easy to see that $\mathbb{E}[|T|] = ps = 2s' \log n$. By applying the standard Chernoff bound,

$$\Pr\left[|T| \geq 5s' \log n\right] \leq \exp\left(-\frac{9s' \log n}{7}\right) \leq n^{-2}.$$

So with high probability, we have that $|T| \leq 5s' \log n$. Further,

$$\mathbb{E}[|G \cap T|] \geq (1-\delta)ps = 2(1-\delta)s' \log n.$$

Again, by applying the standard Chernoff bound,

$$\Pr\left[|G \cap T| \leq s' \log n\right] \leq \exp\left(-\frac{(0.5-\delta)^2 s' \log n}{(1-\delta)}\right) \leq n^{-2}$$

where the last inequality holds since $s' \geq 2/(0.5-\delta)^2$. So we get with high probability $|G \cap T| \geq s' \log n$. Taking a union bound, we have that with a high probability,

$$|T| \leq 5s' \log n \text{ and } |G \cap T| \geq s' \log n.$$

From now on, we continue the proof by assuming the event that $|T| \leq 5s' \log n$ and $|G \cap T| \geq s' \log n$.

Next, we exhaustively try all possible subsets of the set T and run our reconstruction algorithm from Theorem 10 on each such subset, and finally return the set of all these rankings.

Among all these possible subsets, one of them contains only rankings from the set G and is of size at least $s' \log n$. Using this set, the reconstruction algorithm of Theorem 10 reconstructs a ranking π_c such that $d(\pi_c, \pi) \leq \alpha n$.

Now, we have that $s' = \frac{1}{(0.5-\delta)^2} \cdot \frac{50}{(1-\phi)^2}\left(\log \frac{3}{\alpha} + \log \frac{1}{1-\phi}\right)$. Since $|T| \leq 5s' \log n$, we will have at most $n^{5s'} = n^{\mathsf{poly}(\log(1/\alpha), 1/(1-\phi), 1/(0.5-\delta))}$ many subsets. By Remark 11, the reconstruction algorithm will require at most $O(s' n^3 \log n)$ time for each subset, and thus the lemma follows. ◀

▶ **Lemma 16.** *Consider $\phi \in (0,1)$, $\delta \in (0, 0.5)$, a ranking $\pi \in S_n$. Let S be a (π, ϕ, δ)-corrupted sample set (of size at least $2/(1-2\delta)$). Then $\mathsf{OPT}_\delta(S) \geq \phi n - o(n)$ with a high probability.*

Proof. To prove the above lemma, it suffices to show that for any $S' \subseteq S$ where $|S'| \geq (1-\delta)|S|$, $\mathsf{OPT}(S') \geq \phi n - o(n)$.

By definition, the set $S' \cap G$ only contains the rankings drawn from $M(\pi, \phi)$ and is of size at least two. Then by Lemma 12, $\mathsf{OPT}(S' \cap G) \geq \phi n - o(n)$ (with a high probability). By definition, $\mathsf{OPT}(S') \geq \mathsf{OPT}(S' \cap G)$ (since each of the rankings in $S' \setminus G$ can only increase the value of $\mathsf{OPT}(S')$). So we have $\mathsf{OPT}(S') \geq \phi n - o(n)$. ◀

We are now ready to finish the proof of Theorem 13.

Proof of Theorem 13. Given as input a (π, ϕ, δ)-corrupted sample set S, we first apply the algorithm of Lemma 15 to obtain a set of rankings P. We then simply return a ranking $\tilde{\pi} \in P$ that achieves the minimum objective value, i.e., $\tilde{\pi} := \arg\min_{\rho \in P} \mathtt{Obj}_\delta(S, \rho)$. Observe that we can compute $\mathtt{Obj}_\delta(S, \rho)$ (for each $\rho \in P$) naïvely in polynomial-time because $\mathtt{Obj}_\delta(S, \rho) = \mathtt{Obj}(S', \rho)$, where $S' \subseteq S$ is the set of $(1-\delta)|S|$ many nearest rankings (of S) to ρ.

Next, we argue that $\tilde{\pi}$ is a $(1+\varepsilon)$-approximate δ-maximum aggregate ranking with outliers, i.e., $\mathtt{Obj}_\delta(S, \tilde{\pi}) \leq (1+\varepsilon)\mathsf{OPT}_\delta(S)$. We know that there exists some set G of size at least $(1-\delta)|S|$ such that all the rankings in G are drawn from $M(\pi, \phi)$. By Lemma 7, we can assume that for every $\pi_i \in G$, $d(\pi_i, \pi) \leq (1+\lambda)\frac{\phi}{1-\phi} n$ for some $\lambda = o(1)$. Therefore we know that $\mathsf{OPT}(G) \leq (1+\lambda)\frac{\phi}{1-\phi} n$.

Now, consider any $S' \subseteq S$ such that $|S'| \geq (1-\delta)|S|$. Then we must have $|S' \cap G| \geq (1-2\delta)|S|$. Further, from Theorem 5 we have that for the set $S' \cap G$,

$$\mathsf{OPT}(S' \cap G) \geq (1-\lambda)\frac{\phi}{1-\phi}n - \frac{n}{1-\phi}e^{-\frac{(1-2\delta)|S|(1-\phi)^2}{50}}.$$

As argued in the proof of Lemma 16, $\mathsf{OPT}(S') \geq \mathsf{OPT}(S' \cap G)$. Therefore, we have that

$$\begin{aligned}\mathsf{OPT}(G) &\leq (1+\lambda)\frac{\phi}{1-\phi}n \\ &\leq \mathsf{OPT}(S') + 2\lambda\frac{\phi}{1-\phi}n + \frac{n}{1-\phi}e^{-\frac{(1-2\delta)|S|(1-\phi)^2}{50}} \\ &\leq \mathsf{OPT}(S') + \frac{n}{1-\phi}e^{-\frac{(1-2\delta)|S|(1-\phi)^2}{50}} + o(n).\end{aligned}$$

Since the above is true for all $S' \subseteq S$ of size at least $(1-\delta)|S|$, we derive that

$$\mathsf{OPT}(G) \leq \mathsf{OPT}_\delta(S) + \frac{n}{1-\phi}e^{-\frac{(1-2\delta)|S|(1-\phi)^2}{50}} + o(n).$$

By Theorem 5 (considering G as the sample set), we have

$$\begin{aligned}\mathtt{Obj}(G,\pi) &\leq \mathsf{OPT}(G) + \frac{n}{1-\phi}e^{-\frac{|G|(1-\phi)^2}{50}} + o(n) \\ &\leq \mathsf{OPT}_\delta(S) + \frac{n}{1-\phi}e^{-\frac{(1-2\delta)|S|(1-\phi)^2}{50}} + \frac{n}{1-\phi}e^{-\frac{|G|(1-\phi)^2}{50}} + o(n).\end{aligned}$$

Now, set $\alpha = \frac{\varepsilon\phi}{4}$. By Lemma 15, there exists $\pi_c \in P$ such that $d(\pi_c, \pi) \leq \alpha n$. By Lemma 16, we have that $\mathsf{OPT}_\delta(S) \geq \phi n - o(n)$. Consider $|S| \geq \frac{50}{(0.5-\delta)^2(1-\phi)^2}\left(\log\frac{3}{\alpha} + \log\frac{1}{1-\phi}\right)$. By the triangle inequality, we have

$$\begin{aligned}\mathtt{Obj}(G,\pi_c) &\leq \mathsf{OPT}_\delta(S) + 2\alpha n + o(n) \\ &\leq \mathsf{OPT}_\delta(S) + 3\alpha n \\ &\leq (1+\varepsilon)\mathsf{OPT}_\delta(S).\end{aligned}$$

Since $|G| \geq (1-\delta)|S|$, by the construction,

$$\mathtt{Obj}_\delta(S,\tilde{\pi}) \leq \mathtt{Obj}(G,\pi_c) \leq (1+\varepsilon)\mathsf{OPT}_\delta(S).$$

The running time of the algorithm is dominated by the time required to construct the set of rankings P. Therefore the algorithm requires $n^{\mathsf{poly}(\log(1/\varepsilon),1/(1-\phi),1/(0.5-\delta))}$ time. ◀

B A Simple Exact Reconstruction of the Hidden Ranking

From the previous section, we see that if the number of sampled rankings is large enough, we expect that with a large probability, taking the majority will give the correct order of most pairs of symbols in π. We now prove that exact reconstruction is possible with high probability if the sampled set is of size at least $\frac{12(1+\phi)^2}{(1-\phi)^2}\log n$. Further, we describe a simpler algorithm for the reconstruction if this condition is met.

▶ **Theorem 17.** *Consider a $\phi \in (0,1)$ and a $\pi \in S_n$. There exists a deterministic algorithm that, given a set S of size at least $\frac{12(1+\phi)^2}{(1-\phi)^2}\log n$ drawn from $M(\pi,\phi)$, computes π with probability at least $1-\frac{1}{n}$ in time $O(n\log^2 n)$.*

The algorithm constructs a new ranking using any comparison-based algorithm on $[n]$ and returns this as the hidden ranking. When comparing two symbols a and b, we set $a \prec b$ if at least half the sampled rankings rank a before b and set $a \succ b$ otherwise.

We first need to prove an upper bound on the probability that a ranking drawn from the Mallows model has a given pair of symbols inverted in relative order compared to π.

▶ **Lemma 18.** *Consider a ranking ρ drawn from a Mallows model $M(\pi, \phi)$. The probability that for a given pair of symbols (a, b) that $a \prec_\rho b$ and $b \prec_\pi a$ is at most $\frac{\phi}{1+\phi}$.*

Proof. Recall that from Lemma 8, for some $i < j$ and some sampled ranking π_r, we have $\pi(j) \prec_{\pi_r} \pi(i)$ with probability $\frac{\phi^k(\phi^{k+1}-(k+1)\phi+k)}{(1-\phi^k)(1-\phi^{k+1})}$ where $k = j - i$.

This value is smallest at $k = 1$. Substituting $k = 1$, we get the probability of an inversion is at most $\frac{\phi}{1+\phi}$. ◀

We now want to prove that with high probability, for every pair of symbols, the majority of rankings in S have that pair of symbols in the same relative order as π. If this holds, then following the majority when deciding the relative order of each pair of symbols will exactly reconstruct π.

▶ **Lemma 19.** *Let S be a set of at least $\frac{12(1+\phi)^2}{(1-\phi)^2} \log n$ rankings drawn from $M(\pi, \phi)$. Then we have that with probability at least $1 - \frac{1}{n}$, for every pair of symbols (a, b), if $a \prec_\pi b$, then*

$$|\{\pi_r \mid a \prec_{\pi_i} b, \pi_r \in S\}| \geq \frac{|S|}{2}.$$

Proof. Let $s = |S|$ and take any pair of symbols $a, b \in [n]$.

Given a ranking $\pi_r \in S$, let X_r be an indicator random variable that takes on 1 if π_r has an inversion in relative order of a and b compared to π. By Lemma 18, we have that $\mathbb{E}[X_r] \leq \frac{\phi}{1+\phi}$.

Let $X = \sum_{r=1}^{s} X_r$. Then by the linearity of expectation, $\mathbb{E}[X] \leq s\frac{\phi}{1+\phi}$. Applying the standard Hoeffding bound,

$$\begin{aligned}
\Pr\left[X - \mathbb{E}[X] > \left(\frac{1}{2} - \frac{\phi}{1+\phi}\right)s\right] &\leq \exp\left(\frac{-2\left(\left(\frac{1}{2} - \frac{\phi}{1+\phi}\right)s\right)^2}{s}\right) \\
&\leq \exp\left(-2\left(\frac{1}{2} - \frac{\phi}{1+\phi}\right)^2 \frac{12(1+\phi)^2}{(1-\phi)^2}\log n\right) \\
&\leq \exp\left(-3\log n\right) \\
&\leq n^{-3}.
\end{aligned}$$

As there are $n(n-1)/2$ pairs of symbols, taking a union bound over all such pairs gives that with probability at most $1/n$, there exists at least one pair in different pairwise order compared to π. Therefore, with probability at least $1 - 1/n$, we have that every pair of symbols has the same relative order in at least half of the rankings in S and in π. ◀

Finally, the proof of Theorem 17 follows directly from Lemma 19.

Proof of Theorem 17. Consider the execution of the comparison-based sorting algorithm. Consider each time the algorithm compares some pair of symbols (a, b). If $a \prec_\pi b$ and $|\{\pi_i | a \prec_{\pi_i} b\}| \geq \frac{|S|}{2}$, then it will order this pair of symbols identically to π. If this holds for every pair it compares, then it will produce the ranking π at the end of the execution. From Lemma 19, with probability at least $1 - 1/n$, this holds for every pair of symbols, and so the algorithm correctly returns π. ◀

Reinforcement Planning for Effective ε-Optimal Policies in Dense Time with Discontinuities

Léo Henry
University College London, UK

Blaise Genest
CNRS and CNRS@CREATE, IPAL, France
Institute for Infocomm Research (I2R), Singapore

Alexandre Drewery
ENS Rennes, France

Abstract

Lately, the model of (Decision) Stochastic Timed Automata (DSTA) has been proposed, to model those Cyber Physical Systems displaying dense time (physical part), discrete actions and discontinuities such as timeouts (cyber part). The state of the art results on controlling DSTAs are however not ideal: in the case of infinite horizon, optimal controllers do not exist, while for timed bounded behaviors, we do not know how to build such controllers, even ε-optimal ones.

In this paper, we develop a theory of Reinforcement Planning in the setting of DSTAs, for discounted infinite horizon objectives. We show that optimal controllers do exist in general. Further, for DSTAs with 1 clock (which already generalize Continuous Time MDPs with e.g. timeouts), we provide an effective procedure to compute ε-optimal controllers. It is worth noting that we do *not* rely on the discretization of the time space, but consider symbolic representations instead. Evaluation on a DSTA shows that this method can be more efficient. Last, we show on a counterexample that this is the furthest this construction can go, as it cannot be extended to 2 or more clocks.

2012 ACM Subject Classification Theory of computation → Timed and hybrid models

Keywords and phrases reinforcement planning, timed automata, planning

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.13

Related Version *Long Version*: `http://perso.crans.org/genest/HGD23.pdf` [13]

Funding DesCartes: this research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, by ANR-20-CE25-0012 MAVeriQ, by PEPR-AI SAIF and by EPSRC Standard Grant CLeVer (EP/S028641/1).

1 Introduction

Controlling *Cyber Physical Systems* (*CPSs*) is an important task to optimize resources of today's complex systems. It is also challenging, as CPSs often mix discrete control, continuous state space, time and stochasticity, between others. A well-studied model of CPSs is Continuous-Time Markov Decision Processes (*CTMDPs*). On the theoretical side, optimal controllers exist in the timed-bounded horizon setting [3, 16, 8], with infinite horizon [6, 12], or with discounted infinite horizon [7]. Yet, building such optimal controller might not be as simple, even in the time bounded case [15]. On the practical side, there are many algorithms to produce ε-optimal controllers [16, 8]. CTMDPs are fully continuous models, expanded in the previous decade by *Decision Stochastic Timed Automata* (DSTA, see [5, 4]), able to express hard real-time constraints (e.g. timeouts, deadlines) using real-valued clocks and guards on transitions stemming from the *Timed Automata* [1] and *Stochastic TAs* [2] formalisms. In terms of DSTAs, CTMDPs are DSTAs with a single clock reset on all transitions and no guards.

© Léo Henry, Blaise Genest, and Alexandre Drewery;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 13; pp. 13:1–13:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Theoretically, this ability to model hard constraints is very challenging as it produces discontinuities. It breaks the proof for the existence of optimal controller for CTMDPs and to produce ε-optimal policies in practice (except in the special case of value 1 [4]). In [5], an existential theorem for optimal policies for timed bounded horizons in DSTA has been proved in a *non*-constructive manner. However, besides the fact that they can be chosen positional, little is known on the kind of policies necessary to achieve optimality.

In this paper, we develop a *Reinforcement Planning* (*RP*) framework [18] to prove (Theorem 7) the existence of optimal positional policies for DSTA in general for unbounded horizon equipped with a discount. This is the best possible result, as optimal policies do not exist in general for unbounded horizon without discount [5].

Our main results concern ε-approximations: we prove that for one-clock DSTAs, which are already more powerful than CTMDPs (the unique clock needs not be reset at every transition) and allows to model discontinuous hard constraints such as timeouts, *polytope policies* (defined hereafter) suffices. A polytope policy partitions $[0, \infty)$ (value of the clock) with a finite number of intervals and plays one action on each interval. Technically, we develop a symbolic value iteration algorithm, creating polytopes only when necessary using *symbolic functional analysis* (Lemma 18); and then use *numerical analysis* to effectively compute approximations of these polytopes (Lemma 20). Last, we show that this is the furthest this method can go, as in the presence of 2 clocks or more, the optimal policy after n iterations is not a polytope policy (Theorem 9). A long version is available [13].

Related Work. Reinforcement *Planning* [18, 10] provides foundations to reason about observations for completely specified models, that ground Reinforcement *Learning* (e.g. Monte-Carlo method) which tackle un- or semi-specified systems. RP provides implementable algorithms for fully specified models, which is also the case of our methodology, although the implementation would require a heavy use of both symbolic functional as well as numerical analysis tools which are very technical and tedious to implement. RP has first been theorized for discrete models with finite state space (MDPs) [18]. Some version exist for a totally continuous world (continuous state space, continuous action, continuous behavior) [10]. Some RP frameworks have been developed for some specific model with discrete actions and continuous state space: CTMDPs [7], ODEs [19], but without any discontinuous behaviors such as deadline or timeouts. We did not find any RP methodology which can be applied to a setting mixing continuous state space, discrete action set and non-continuous behaviors (timeouts, deadlines). The closest related work is [9], where the authors design an effective algorithm to produce ε-optimal policies. The main differences are that the method is very different, discretizing time in many modes of ϵ-size (depending on the ε-precision desired), whereas we compute a minimal number of (larger) modes using symbolic function analysis; and the model considered is different, namely Markov Timed Automata (MTA). Compared with DSTA, MTA cannot model urgency: time can always elapse, and all distributions are exponential, whereas DSTA can enforce leaving a state before some time-out using uniform distributions. Further, uniform distributions are harder to handle than exponential ones. It is thus unclear how to lift similar time-discretization bounds for DSTA. We numerically evaluate on a DSTA the efficiency of our technique in Section 5.3 and compare with the complexity of time-discretization for a MTA of similar size.

2 Decision Stochastic Timed Automata

We start by defining the model of CPS we are considering in the paper, namely Decision Stochastic Timed Automata (*DSTA* for short), extending CTMDPs with hard constraints.

2.1 Notations

We fix finite sets $\mathcal{C}$ of clocks, and $\mathcal{V}$ of *valuations over* $\mathcal{C}$, i.e. the set of all maps of $\mathcal{C} \to \mathbb{R}_{\geq 0}$ (clocks always have non-negative values). Given a valuation $v \in \mathcal{V}$, we define two operations:
Time elapse $v + \delta$ with $\delta \in \mathbb{R}_{\geq 0}$ is a valuation such that for all $x \in \mathcal{C}$ $(v+\delta)(x) = v(x) + \delta$;
Reset $\mathcal{C}' \subseteq \mathcal{C}$ $v_{[\mathcal{C}' \leftarrow 0]}$ is the valuation $x \mapsto 0$ if $x \in \mathcal{C}'$ and $x \mapsto v(x)$ otherwise.
We will sometimes consider a valuation v as a vector in $\mathbb{R}_{\geq 0}^{|\mathcal{C}|}$.

We denote $\mathcal{G}$ the set of guards, i.e. of constraints of the form $\bigwedge_{1 \leq i \leq |\mathcal{C}|}(x_i \in I_i)$ and $\mathcal{I}$ the set of *invariants*, i.e. guards featuring intervals with no lower bound. The set of *constraints over a set of clocks* $\mathcal{C}$ is the set of all constraints of the form $(\sum_{1 \leq i \leq |\mathcal{C}|} a_i \times x_i) \bowtie q$ with $a_i \in \mathbb{Q}$, $x_i \in \mathcal{C}$, I_i an interval of $\mathbb{Q}$, $\bowtie \in \{<, \leq, =, \geq, >\}$ and $q \in \mathbb{Q}$. A *polytope* $z \in \mathcal{Z}$ is a finite set of constraints over $\mathcal{C}$. Let $z \in \mathcal{Z}$ and $v \in \mathcal{V}$. We write $v \models z$ if and only if v satisfies all constraints in z. A polytope z defines a subset $\mathcal{V}(z)$ of $\mathbb{R}^{|\mathcal{C}|}$ containing the valuations v such that $v \models z$. We define $z_{[\mathcal{C}' \leftarrow 0]}$ as the polytope such that $\mathcal{V}(z_{[\mathcal{C}' \leftarrow 0]}) = \{v_{[\mathcal{C}' \leftarrow 0]} \mid v \in \mathcal{V}(z)\}$.

For a function P from some set to booleans we note $1_P : y \mapsto 1$ iff $P(y)$ and otherwise 0. We write $\mathcal{D}(Y)$ for the set of all probabilities distributions over a set Y.

2.2 Decision Stochastic Timed Automata

We consider DSTAs [5, 4], transition systems defining probabilities on sets of runs by assigning probability distributions to delays, up to a policy dictating the transitions taken.

▶ **Definition 1.** *A DSTA over clocks $\mathcal{C}$ and actions A is a tuple $\mathcal{A} = (\mathcal{L}, E, \ell_0, \mathrm{I}, \mu)$ with*
- *$\mathcal{L}$ a set of control state and $\ell_0 \in \mathcal{L}$ is the initial location,*
- *E a set of transitions of the form $(\ell_1, a, g, r, \ell_2) \in \mathcal{L} \times A \times \mathcal{G} \times 2^{\mathcal{C}} \times \mathcal{L}$, where g is the guard of the transition and r is the finite set of clocks to reset when taking the transition,*
- *$\mathrm{I} : \mathcal{L} \to \mathcal{I}$ defines the invariants the clock values need to satisfy to stay in a control state,*
- *$\mu = (\mu_{(\ell,v)})_{(\ell,v) \in L \times V(X)}$ defines probability distributions over delays for each control state and clock valuation. For $\mathrm{I}(\ell, v) = \{\delta \in \mathbb{R}^+ | v + \delta \models \mathrm{I}(\ell)\}$, it is of one of the following type:*
  - *if $\mathrm{I}(\ell, v)$ is the singleton $\{0\}$, then μ is the Dirac distribution $\mu(0) = 1$,*
  - *otherwise, if $\mathrm{I}(\ell, v)$ is bounded, then μ is the uniform distribution over $\mathrm{I}(\ell, v)$,*
  - *otherwise, $\mathrm{I}(\ell, v) = [0, \infty)$ and μ is an exponential distribution $\mu(\delta) = e^{-\alpha_\ell \delta}$ for some parameter $\alpha_\ell \in \mathbb{R}_{>0}$ independent over the valuation v.*

Intuitively, a move in a DSTA $\mathcal{A}$ from a configuration $(\ell, v) \in \mathcal{L} \times \mathcal{V}$ is done in two steps: first, a delay δ is randomly chosen according to the distribution $\mu_{(\ell,v)}$, and then the policy chooses a transition $e = (\ell, a, g, r, \ell')$ enabled from $(\ell, v + \delta)$, reaching $(\ell', (v + \delta)_{[r \leftarrow 0]})$.

Formally, we define a *partial run* of a DSTA as a sequence of transitions and delays $\rho = (e_i, \delta_i)_{1 \leq i \leq n}$ starting from a configuration (ℓ_1, v_1) such that a corresponding sequence of configurations can be defined: $\rho = \big((\ell_i, v_i) \xrightarrow{e_i} (\ell_{i+1}, v_{i[r_i \leftarrow 0]}) \xrightarrow{\delta_i} (\ell_{i+1}, v_{i[r_i \leftarrow 0]} + \delta_i)\big)_{1 \leq i \leq n}$ (notably, the transitions e_i are enabled in (ℓ_i, v_i)). A *run* is a partial run starting in the initial configuration $(\ell_0, \mathbf{0})$. We note $Runs(\mathcal{A})$ (resp. $PRuns(\mathcal{A}, (\ell, v))$) the set of runs (resp. partial runs) of a DSTA $\mathcal{A}$ (resp. starting from a configuration (ℓ, v)).

To start with a random passing of time, we can add a dummy state **init** with no incoming transition. Every action from **init** leads to ℓ_0, resetting all clocks, so that any random delay can be taken in **init**. We will not depict such a state **init** in the different figures.

▶ **Example 2.** Figure 1 depicts DSTA $\mathcal{A}$. In $\mathcal{A}$, the user can play b in ℓ_0 to roll for a new clock value. When they play a, they are sent to ℓ_1. Depending on the delay randomly chosen in ℓ_1, the control states either goes back to ℓ_0 ($x > 1$) and resets the clock x, or it moves to

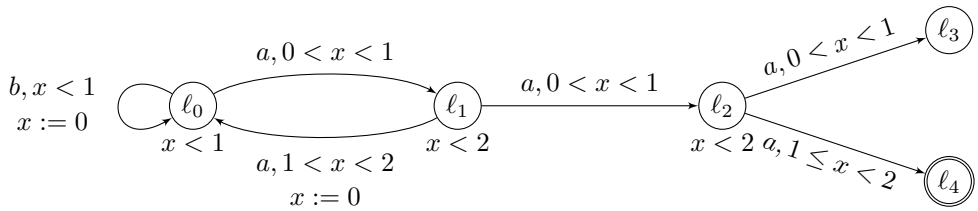


Figure 1 A DSTA with one clock x with initial state ℓ_0. The goal is to reach ℓ_4.

ℓ_2 ($x < 1$) from which another delay is randomly drawn and added to x: if this delay makes $x > 1$, then the run goes to winning state, else it goes to a losing state. Intuitively, playing a from ℓ_0 with x very close to 1 is optimal, as then the chance to get stuck in the losing state is very small (the sum of the delays from ℓ_1 and ℓ_2 needs to be smaller than $1 - x$).

To discuss the semantics of DSTAs, we have to define the probabilities some behaviors have to appear. As (almost) every run has a null probability to appear, we discuss at the level of (partial) *paths*, i.e. sequences of transitions. A path $\pi = (e_i)_{1 \leq i \leq n}$ represents the set of runs ρ such that there exist delays $(\delta_i)_{1 \leq i \leq n}$ with a run $\rho = (e_i, \delta_i)_{1 \leq i \leq n}$. We say that a run ρ *models* a path $\pi \in E^*$ when their sequences of transitions match. A partial path is said *feasible* when it has a model. We call *path* a partial path that is modeled by a (initial) run. For paths, we will only consider (initial) runs as models. We write $PPaths(\mathcal{A})$ (resp. $Paths(\mathcal{A})$) for the set of partial paths (resp. paths) of an automaton $\mathcal{A}$.

2.3 Rewards and policies

We consider a discounted average reward model as the objective function (as usual in Reinforcement Planning), by contrast with winning state and infinite horizon (reward= 1 if winning state ever reached, 0 otherwise) or finite horizon (reward = 1 if winning state reached before the time horizon, 0 otherwise). More precisely, each transition is associated with a reward, and we consider the *discounted sum* of all rewards as our objective function, with immediate rewards of greater interest than later ones. Discount allows to tackle unbounded runs, while optimal policies do not exist for unbounded horizons without discount [5].

▶ **Definition 3.** *Consider a DSTA $\mathcal{A} = (\mathcal{L}, E, \ell_0, \mathrm{I}, \mu)$. A* reward function *is a function $\mathcal{R} : E \to \mathbb{N}$ associating a reward to each transition. Given discount $0 < \gamma < 1$, the* discounted reward *of a run $\rho = (e_i, v_i)_i$ or of a path $\pi = (e_i)_i$ is $\mathcal{R}(\rho) = \mathcal{R}(\pi) = \sum_i \gamma^{i-1}\mathcal{R}(e_i)$.*

The rewards associated with Figure 1 are 0 on every transition except on from ℓ_2 to ℓ_4, where the reward is 1. Hence a path reaching ℓ_4 in i steps will get discounted reward γ^i.

The decisions made by the agent are formalized as *policies*. Policies depend on the type of input they can use to take a decision: *generic policies* can use any information about the history of the run to make stochastic choices, while more restrictive *pure positional policies* can only use the current configuration and decide on a given transition deterministically.

▶ **Definition 4.** *A policy σ on a DSTA $\mathcal{A} = (\mathcal{L}, E, \ell_0, \mathrm{I}, \mu)$ is a function $\sigma : Runs(\mathcal{A}) \to \mathcal{D}(E)$. A pure positional policy is a policy σ such that for all pairs of runs ρ, ρ' ending in the same configuration $(\ell, v) \in \mathcal{L} \times \mathcal{V}$, we have $\sigma(\rho) = \sigma(\rho')$ is a Dirac distribution in one action. We can thus represent any pure positional policy as a function $\sigma : (\mathcal{L} \times \mathcal{V}) \to E$.*

Given a policy σ and a history $\rho \in Runs(\mathcal{A})$ ending in (ℓ, v), we can define the probability $P_\sigma(\pi \mid \rho)$ for the behavior of the DSTA to model the path $\pi = (e_i)_{1 \leq i \leq n}$ after ρ as follows:

$$P_\sigma(\pi \mid \rho) = \sigma(\rho)(e_1) \int_{\delta \in \mathrm{I}(\ell_1, v_{[r_1 \leftarrow 0]})} P_\sigma((e_i)_{2 \leq i \leq n} \mid \rho \cdot (e_1, \delta)) d\mu_{(\ell_1, v_{[r_1 \leftarrow 0]})}$$

with ℓ_1 the target location of e_1 and r_1 the set of clocks it resets. Given a reward function $\mathcal{R}$ this can be used to assign a *value* to a policy.

▶ **Definition 5.** *Fix a reward function $\mathcal{R}$ on a DSTA $\mathcal{A}$ with discount γ and an integer $n \in \mathbb{N}$. The value of a policy σ after a history $\rho \in Runs(\mathcal{A})$ for n steps is*

$$Val^n_\sigma(\rho) = \sum_{\pi = (e_i)_{1 \leq i \leq n} \in E^n} P_\sigma(\pi \mid \rho) \mathcal{R}(\pi) \ .$$

The value of a policy *for history ρ is $Val_\sigma(\rho) = \lim_{n \to \infty} Val^n_\sigma(\rho)$. We write $Val^*(\rho) = \sup_\sigma Val_\sigma(\rho)$ for the optimal value. The value $Val^{(*)}_{(\sigma)}(\mathcal{A}) = Val^{(*)}_{(\sigma)}(\lambda)$, denoted $Val^{(*)}_{(\sigma)}$, is the value associated with the empty run. A policy σ^* is said* optimal *when $Val^* = Val_{\sigma^*}$.*

Rewards and values are bounded with known bounds. We write MaxReward (resp. MaxValue) for the supremal reward (resp. value). Similarly, we adopt the notation $\mathrm{Val}_\sigma(\rho, e)$ (resp. $\mathrm{Val}^*(\rho, e)$) to express the value of first playing e and then following σ (resp. having an optimal value). Finally, for (pure) positional policy, we replace ρ by a configuration (ℓ, v) (e.g. $\mathrm{Val}_\sigma(\ell, v)$). We say that a policy σ is *ε-optimal* whenever $\mathrm{Val}_\sigma(\mathcal{A}) \geq \mathrm{Val}^*(\mathcal{A}) - \varepsilon$.

In general, one cannot represent finitely a measurable policy, nor even a pure positional policy even for a DSTA with a unique clock. We now define a class of policies that can be finitely described, called *polytopes policies* [5]. Given a control state ℓ, it partitions $\mathcal{V}$ with a finite number of polytopes and assigns a unique action to each polytope.

▶ **Definition 6.** *Let $\mathcal{A}$ be a DSTA. A pure positional policy σ is a* polytope policy *if for each location ℓ of $\mathcal{A}$ there exists a finite set of polytopes $Z_1, \ldots, Z_n$ that partition $\mathcal{V}$ such that $\forall 1 \leq i \leq n,\ \forall v, v' \in Z_i,\ \sigma(\ell, v) = \sigma(\ell, v')$.*

3 Main Results

Our first contribution is to show, by adapting the RP framework to DSTA (see Section 4, which serves as proof of Theorem 7), that *pure positional* optimal policies exist:

▶ **Theorem 7.** *Let $\mathcal{A}$ be a DSTA. Then there exists a pure positional policy σ^* such that for every policy σ and every run ρ, $Val_{\sigma^*}(\rho) \geq Val_\sigma(\rho)$.*

Notice that discounted infinite horizon is the most generic reward scheme possible enjoying the existence of optimal policies, as it has been shown in [5] that optimal policies do not exist in general for undiscounted infinite horizon. This matches the existence of optimal policies for finite horizon properties of [5], with a vastly different proof technique.

Concerning ε-optimal policies, we show that the class of *polytope policies*, which are finitely describable pure positional policies, are sufficient for DSTA with a unique clock $\mathcal{C} = \{x\}$. The main difficulty is that there are *uncountably many* configurations, and from each a (pure positional) policy needs to provide which action to play. We do so by developing a *symbolic* value iteration algorithm and prove in Lemma 18 that the policies generated are polytope policies, using symbolic functional analysis. Even harder is to actually

compute effectively a ε-approximation from the *uncountably many* configurations, and not only dealing with existence: Using numerical functional analysis, we explain how to compute an approximation of the finitely many positions of the changes of sign of the value function, without approximating the value function itself. This is our second and main contribution:

▶ **Theorem 8.** *Let $\mathcal{A}$ be a DSTA with a unique clock and $\varepsilon > 0$. Then there exists an effectively constructible positional polytope policy σ_ε that is ε-optimal.*

Section 5 serves as proof of Theorem 8: Lemma 18 shows that there exist an *optimal* polytope policy σ_n *for n steps of value iteration.* This is complex as we do not know how to provide a close-form solution for the value function itself, and we conjecture that it is a hard problem. Eventually, after n iterations, σ_n will be $\frac{\varepsilon}{2}$-optimal. Using numerical analysis, we show in Lemma 20 in Section 5 how to compute effectively a polytope policy σ whose value is at most $\frac{\varepsilon}{2}$ away from the value of σ_n, providing us with a polytope policy that is ε-optimal. This is in stark contrast with the non-constructive proof of [5], which needs uncountable time to produce ε-optimal policies.

According to Lemma 18, polytope policies are optimal for DSTA with 1 clock and without loops, as the value is met after a finite number of steps of value iteration. Our third contribution is that polytope policies are not optimal with 2 clocks and without loops.

▶ **Theorem 9.** *There is a DSTA $\mathcal{A}$ with 2 clocks and without loops such that for all polytope policies σ, there exists a policy σ' with $Val_{R,\gamma}(\mathcal{A},\sigma) < Val_{\mathcal{R},\gamma}(\mathcal{A},\sigma')$.*

The proof can be found in Section 5.7. It is inspired by [5], where it is shown polytope policies are not optimal for finite horizon properties, even for DSTA with a *unique* clock. We use the second clock to simulate the bound on the time horizon. Theorem 8 shows that for one clock, the setting of discounted infinite horizon has much more desirable properties than the one of bounded time. Notice that polytope policies could still be ε-optimal in general.

4 Bellman equations in dense time

We extend Reinforcement Planning [18] to DSTA models. We first focus on the introduction of Bellman Equations for this continuous setting. The Bellman Equations are the theoretical basis of RP, as they allow to incrementally compute or approximate the (optimal) values.

4.1 General policies

We first reason on general policies in order to prove the optimality of the *pure positional policies*. Once their optimality is proved, the discussion is restricted to this simpler class. We first define the Bellman Equations in their most general form for DSTA.

▶ **Proposition 10.** *Given a policy σ and an history ρ ending in configuration (ℓ, v), $Val_\sigma(\rho)$ is the only bounded solution of the* **Bellman Equation in dense time***:*

$$Val_\sigma(\rho) = \sum_{e=(\ell,a,g,r,\ell')\in E} \sigma(\rho)(e) \times \Big(\mathcal{R}(e) + \gamma \times \int_{\delta\in \mathrm{I}(\ell',v_{[r\leftarrow 0]})} Val_\sigma(\rho\cdot(e,\delta))d\mu_{(\ell',v_{[r\leftarrow 0]})}\Big).$$

In a similar fashion, $Val_\sigma(\rho, e)$ for $e = (\ell, a, g, r, \ell')$ is the unique solution of:

$$Val_\sigma(\rho, e) = \mathcal{R}(e) + \gamma \times \int_{\delta\in \mathrm{I}(\ell',v_{[r\leftarrow 0]})} \sum_{e'=(\ell',a,g,r,\ell'')\in E} \sigma(\rho)(e') \times Val_\sigma(\rho\cdot(e,\delta), e')d\mu_{(\ell'',v_{[r\leftarrow 0]})}.$$

Proof sketch. We prove that the value is a solution by using the definition of Val as the limit of Val^n and unfolding it (taking the first action out of the limit). The uniqueness of the bounded solution is proved by comparing two potentially different one: $F(\rho) - F'(\rho)$. Using the equation allows to reduce their difference step by step as the first reward does not depend on F / F'. After n steps, the difference is bounded by γ^n times a probability distribution of differences. As the functions are bounded, taking the limit proves the uniqueness. ◀

From Bellman's equations, the following relations can be inferred:

▶ **Corollary 11.** *Given a policy σ and a run ρ ending in (ℓ, v):*

$$Val_\sigma(\rho) = \sum_{e \in E} \sigma(\rho)(e).Val_\sigma(\rho, e) \quad \textbf{and} \quad Val_\sigma(\rho, e) = \mathcal{R}(e) + \gamma \int\limits_{\delta \in \mathrm{I}(\ell', v_{[r \leftarrow 0]})} Val_\sigma(\rho(e, \delta)) d\mu_{(l', v_{[r \leftarrow 0]})}$$

with r the reset of e and l' its target.

For any policy, we can construct a pure positional policy at least as efficient:

▶ **Proposition 12.** *For any policy σ, there exists a pure positional policy σ' such that $Val_{\sigma'} \geq Val_\sigma$. In particular one such policy can be defined from σ in the following way: consider a history ρ ending in configuration (ℓ, v), then*

$$\sigma'(\ell, v) = \arg\max_{e \in E} \sup_{\rho' \text{ ending in } (\ell, v)} \left(\mathcal{R}(e) + \gamma \int\limits_{\delta \in \mathrm{I}(l', v_{[r \leftarrow 0]})} Val_\sigma(\rho' \cdot (e, \delta)) d\mu_{(\ell, v_{[r \leftarrow 0]})} \right).$$

Proof sketch. Using Bellman Equation on Val_σ allows to see that it is lower than what is obtained by playing optimally at the next step (on all executions) and then following σ. Applying the same reasoning n times allows to show that it is lower than $\mathrm{Val}^n_{\sigma'}$ plus a bounded term depending of Val_σ discounted by γ^n. Taking the limit then allows to conclude. ◀

4.2 Positional policies

As we now know from Proposition 12 that positional policies are as efficient as general policies, the rest of this paper mentions only pure positional policies. We will use the Bellman Equation in their restricted form for pure positional policies, provided now:

$$\mathrm{Val}_\sigma(\ell, v) = \mathcal{R}(\sigma(\ell, v)) + \gamma \int\limits_{\delta \in \mathrm{I}(\ell', v_{[r \leftarrow 0]})} \mathrm{Val}_\sigma(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})}$$

with ℓ' the target location of $\sigma(\ell, v)$.

$$\mathrm{Val}_\sigma(\ell, v, e) = \mathcal{R}(e) + \gamma \int\limits_{\delta \in \mathrm{I}(\ell', v_{[r \leftarrow 0]})} \mathrm{Val}_\sigma(\ell', v_{[r \leftarrow 0]} + \delta, \sigma(\ell', v_{[r \leftarrow 0]} + \delta)) d\mu_{(\ell', v_{[r \leftarrow 0]})}$$

with $e(l, x, a) = (l, a, g, r, l')$. The Bellman Equations rely on the integral of the value on the interval of possible delays. We call this the *interval value* function, and write $\mathrm{IVal}_\sigma(\ell, v) = \int_{\delta \in \mathrm{I}(\ell', v)} \mathrm{Val}_\sigma(\ell', v + \delta) d\mu_{(\ell', v)}$. We have $\mathrm{Val}_\sigma = \mathcal{R}(\sigma(\ell, v)) + \gamma.\mathrm{IVal}_\sigma(\ell', v_{[r \leftarrow 0]})$.

We can show that if locally deviating from a value function according to a policy always gives a better result, then globally changing to the new policy enhances the expected return.

▶ **Proposition 13.** *Consider a function F such that for all configurations (ℓ, v) $F(\ell, v, e) = \mathcal{R}(e) + \gamma \int_{\delta \in \mathrm{I}(\ell', v_{[r \leftarrow 0]})} F(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})}$ with ℓ' the target of e. For a policy σ, if for all configurations (ℓ, v), $F(\ell, v, \sigma(\ell, v)) \geq F(\ell, v)$ then $Val_\sigma \geq F$. Furthermore $Val_\sigma > F$ if and only if there exists a configuration where the inequality is strict.*

Proof sketch. The result is obtained as previously by unfolding the equation for n step before using the discount and the boundedness of functions to conclude. ◀

Corollary 16 apply this to obtain a greedy improvement from the current policy. We now state the *Bellman optimality equations in dense time*, characterizing the optimal values.

▶ **Proposition 14.** *Val^* is the only bounded solution of the **Bellman Optimality Equation**:*

$$Val^*(\ell, v) = \max_{e \in E} \left(\mathcal{R}(e) + \gamma \int_{\delta \in \mathrm{I}(\ell', v_{[r \leftarrow 0]})} Val^*(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \right)$$

with r the resets of e and ℓ' its target location. Similarly:

$$Val^*(\ell, v, e) = \mathcal{R}(e) + \gamma \int_{\delta \in \mathrm{I}(\ell', v_{[r \leftarrow 0]})} \max_{e \in E} Val^*(\ell, v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \ .$$

Proof sketch. The proof goes as for Bellman Equations: unfolding the definition of Val^* on one step allows to prove that it is a solution, while uniqueness is showed by comparing two solutions and using γ to limit their difference to 0 in the limit. ◀

We will again write $\mathrm{IVal}^*(\ell, val) = \int_{\delta \in \mathrm{I}(\ell, v)} \mathrm{Val}^*(\ell', v + \delta) d\mu_{(\ell', v)}$. As for values of policies, we use it to deduce relations between optimal values for configurations and transitions.

▶ **Corollary 15.** *For any configuration (ℓ, v) and transition $e \in E$ we have the following equalities: $Val^*(\ell, v) = \max_{e \in E} Val^*(\ell, v, e)$ and $Val^*(\ell, v, e) = \mathcal{R}(e) + IVal^*(\ell', v_{[r \leftarrow 0]})$*

Using these equations and Proposition 13, we obtain two important results: the policy improvement theory works as for MDPs (it has no local minima) and the pure positional policies are optimal.

▶ **Corollary 16.** *Given a policy σ, we can deduce a positional pure policy σ' such that $\forall (\ell, v) \in \mathcal{L} \times \mathcal{V}$, $\sigma'(\ell, v) = \arg\max_{e \in E} \left(\mathcal{R}(e) + \gamma . IVal_\sigma(\ell', v_{[r \leftarrow 0]}) \right)$ with r the reset of e and l' its target. Then $Val_{\sigma'} \geq Val_\sigma$. Furthermore, if $Val_{\sigma'} = Val_\sigma$ then σ is optimal.*

We now restate Theorem 7 in this context and prove it:

▶ **Theorem 17.** *There exists a pure positional policy σ^* such that $Val_{\sigma^*} = Val^*$.*

Proof sketch. For each configuration, we consider the action which is associated with the maximal expected reward when being played from this configuration (chosen arbitrarily from the actions with the same expected maximal reward). We call such an action an optimal action. We consider the positional policy which plays these optimal actions from each configuration. Using Proposition 13, we can show that such a policy is indeed optimal. ◀

Notice that the above proofs crucially rely on the fact that the discount $\gamma < 1$. Indeed, for $\gamma = 1$ (infinite horizon reachability *without* discount), playing positionally the optimal action may lead to very bad outcome. E.g., in Figure 1, the optimal action from $(\ell_0, x < 1)$ is to play action b, looping forever in location ℓ_0, as the expected reward associated with playing b is higher (the supremal value is 1) than the reward from playing a which has some risk associated with it (of value $1 - \varepsilon$). When there is a discount $\gamma < 1$, playing a becomes optimal from $(\ell_0, x < 1)$ for x close to 1, as waiting for a better value of x is not worth it.

5 Computing ε-optimal Policies

5.1 Mode Policies for DSTA with 1 clock

We now restrict ourselves to DSTA with a single clock x. 1-clock polytopes are intervals (called modes), i.e. they can be represented by two bounds in $\mathbb{R}_{\geq 0} \cup \{+\infty\}$. We call *mode policies* such polytope policies. A mode policy σ of DSTA $\mathcal{A}$ with control state $\mathcal{L}$ is represented in the following finite way: $\sigma = (\mathcal{M}_\sigma^\ell, \sigma^\ell)_{\ell \in \mathcal{L}}$ with:

- $\mathcal{M}_\sigma^\ell$ is a partition of $\mathbb{R}_{\geq 0}$ in a finite set of modes $M_1 \dots M_k \in \mathcal{M}_\sigma^\ell$ (the modes of ℓ),
- For all $\ell \in \mathcal{L}$, we have $\sigma^\ell : \mathcal{M}_\sigma^\ell \to E$ gives the action associated with each mode of ℓ.

We will represent the bounds of the interval with rational numbers in $\mathbb{Q}$, wlog as we only need ε-approximations. Mode-policies allow for a finite number of points of discontinuity.

Closeness or openness of intervals is not a concern as there is probability 0 to reach the exact boundary *between two modes* in a control state. What happens at b for the maximal mode $[a, b]$ of a control state might be important though (e.g. with $b = a$ caused by an invariant), so we have one special mode at the end of the interval for $[b, b]$ whenever $b < +\infty$.

5.2 Symbolic Value Iteration for mode policies

We proceed in two steps: First, we provide our symbolic value iteration Algorithm 1 to define the exact mode policy σ_i^* obtained after i steps of the value iteration. Proposition 19 shows that the value Val_i^* of σ_i^* converges towards the optimal value, and given an $\varepsilon > 0$, it provides an effectively computable i such that $\mathrm{Val}_i^* > \mathrm{Val}^* - \frac{\varepsilon}{2}$. Lemma 18 proves that σ_i^* has a finite number of modes, for all $i \in \mathbb{N}$. We never compute the value function itself (we do not know how to represent it). Further, modes, e.g. the exact value of the (possibly non-rational) boundaries, cannot be computed perfectly.

The second step handles these issues, by computing effectively a mode policy $\overline{\sigma_1}$ approximating σ_1^*, such that its value is at least $\overline{\mathrm{Val}_1} > \mathrm{Val}_1^* - \varepsilon_1$. We set ε_1 sufficiently small such that inductively, we obtain a policy $\overline{\sigma_i}$ with value $\overline{\mathrm{Val}_i} > \mathrm{Val}_i^* - \frac{\varepsilon}{2}$. Overall, $\overline{\mathrm{Val}_i} > \mathrm{Val}^* - \varepsilon$.

To ease the presentation, we assume that only two actions $\{\alpha, \beta\}$ are available. It is easy to extend to more actions by comparing actions pairwise. We focus on modes where both α, β are allowed, as the policy is trivial if only one action is allowed. On these modes, Val is continuous. Recall that for $\mathrm{I}(\ell) = [a, b]$, we denote $\mathrm{IVal}(\ell, x) = \int_x^b \mathrm{Val}(\ell, t) d\mu(\ell, t)$.

Step 1: Symbolic Function Analysis to define the exact mode policy

The structure of Algorithm 1 is as follows: from initial value $\mathrm{Val}_0 = 0$, we inductively define the next action value using Bellman Optimality (line 7). Then, the modes of this new value function are defined (line 9) and for each mode the best action is chosen (line 13). Finally the new value function is *defined* for each mode (line 14). Notice that we only need to *determine* the boundaries where the sign of $\mathrm{Val}_i(\ell, x, \alpha) - \mathrm{Val}_i(\ell, x, \beta)$ changes, we do not need to *compute* or *represent* the value function (*symbolic* nature). The algorithm relies on the crucial fact that the number of such boundaries at each step is finite (Lemma 18).

▶ **Lemma 18.** *Let $\ell \in \mathcal{L}$ a control state. For all $i \in \mathbb{N}$, defining $f : x \mapsto \mathrm{Val}_i(\ell, x, \alpha) - \mathrm{Val}_i(\ell, x, \beta)$, there is a $m \in \mathbb{N}$ such that for all $j \leq m$, there exists a sequence $(\tilde{f}^{(j)})_{j=0}^m$ of derivable functions, called the pseudo-derivatives of f, with:*

- *the sign of $\tilde{f}^{(0)}$ and of f coincide in all points of M,*
- *for $1 \leq j \leq m$, the sign of $\tilde{f}^{(j)}$ and of $\frac{d\tilde{f}^{(j-1)}}{dt}$ coincide in all points of M, and*
- *function $\tilde{f}^{(m)}$ has a constant sign over M.*

In particular, functions $f, \tilde{f}^{(0)}$ change sign at most m times.

Algorithm 1 Symbolic Value Iteration Algorithm to define the exact mode policy σ_i^*.

1 Init: $i := 0$ and $\forall \ell \in \mathcal{L}, x \in \mathbb{R}_{\geq 0}, \iota \in \{\alpha, \beta\}$, let $\mathrm{Val}_0^*(\ell, x) := \mathrm{Val}_0^*(\ell, x, \iota) = 0$;
2 σ_0^* has only one mode $M = [0, \infty)$ for each $\ell \in \mathcal{L}$ playing ${\sigma_0^*}^\ell(M) = \alpha$;
3 **repeat**
4 **for** *each location* $\ell \in \mathcal{L}$ **do**
5 **for** *each mode* $M \in \mathcal{M}_{\sigma_i}^\ell$ **do**
6 **for** *each action* $\iota \in \{\alpha, \beta\}$, *let* ℓ' *the destination of the transition* ι *and* $r = 1$ *if the clock is reset,* $r = 0$ *otherwise* **do**
7 Define $\mathrm{Val}_{i+1}(\ell, x, \iota) := \mathcal{R}(\iota) + \gamma \mathrm{IVal}_i^*(\ell', (1-r)x)$ for any x;
8 **end**
9 Compute a minimal finite set $\mathcal{M}_M^\ell$ of modes such that on each mode $\mathrm{Val}_{i+1}(\ell, v, \alpha) - \mathrm{Val}_{i+1}(\ell, v, \beta)$ does not change sign;
10 **end**
11 Set $\mathcal{M}_{i+1}^\ell := \bigcup_{M \in \mathcal{M}_{\sigma_i}^\ell} \mathcal{M}_M^\ell$
12 **end**
13 Assign $\sigma_{i+1}^* := (\mathcal{M}_{i+1}^\ell, \sigma_{i+1}^\ell)_{\ell \in \mathcal{L}}$ where for each control state $\ell \in \mathcal{L}$ and mode $M \in \mathcal{M}_{i+1}$, the policy is ${\sigma_{i+1}^*}^\ell(M) = \alpha$ if $\mathrm{Val}_{i+1}(\ell, x, \alpha) - \mathrm{Val}_{i+1}(\ell, x, \beta) \geq 0$ for $x \in M$, and ${\sigma_{i+1}^*}^\ell(M) = \beta$ otherwise;
14 Define $\mathrm{Val}_{i+1}^*(\ell, v) := \mathrm{Val}_{i+1}(\ell, v, \sigma_{i+1}^\ell(M))$ for $v \in M$;
15 Let $i := i + 1$;
16 **end**

Line 13 computes the sign of $\mathrm{Val}_{i+1}(\ell, x, \alpha) - \mathrm{Val}_{i+1}(\ell, x, \beta) = f(x)$. As f changes sign at most m times (Lemma 18), we have at most $m + 1$ modes. The proof of Lemma 18 can be found in Section 5.4 and relies on the symbolic derivation of $\tilde{f}^{(j)}$ depending on the probability distributions $d\mu(\ell, t)$. We can now prove that Algorithm 1 produces an ε-policy:

▶ **Proposition 19.** *For all $\varepsilon > 0$, Algorithm 1 produces a mode-policy σ_ε^* that is ε-optimal. Further, one can effectively compute a step i after which $(Val^* - \varepsilon) < Val_{\sigma_i^*} \leq Val^*$.*

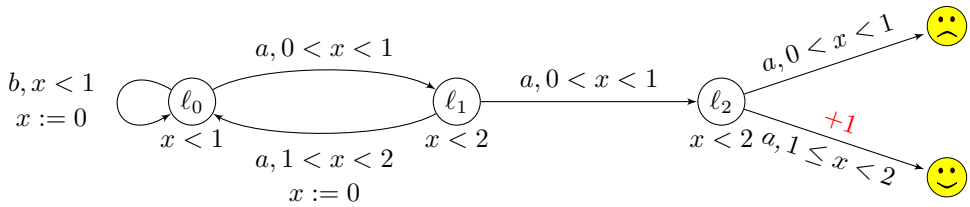
The proof of Proposition 19 can be found in Section 5.5. It is done in three steps: first show that Val_i^* converges, towards a value that is solution of the Bellman Optimality Equation. The third part of the proof provides the number of steps i to ensure a given ε.

To prove Theorem 8, we make symbolic Algorithm 1 *effective*, computing a mode policy $\overline{\sigma_i^*}$ approximating σ_i^*. In particular, we effectively approximate constants coming from integral computations in $(\tilde{f}^{(j)})_{j=0}^m$, then approximate their roots, to obtain approximated roots of f, giving $\mathbb{Q}$-boundaries arbitrarily close to the boundaries of modes of the symbolic σ_i^*.

Step 2: Numerical Analysis to effectively compute an approximated mode policy $\overline{\sigma_i^*}$

Line 13 of Algorithm 1 asks to find the change of sign (or equivalently the roots) for the function f defined in Lemma 18. While the values of the roots may not be easily computable symbolically, we show in the following Lemma that we can approximate them with arbitrarily high precision using numerical analysis (e.g. using Ridder's method [17]):

▶ **Lemma 20.** *Let f and m be as defined in Lemma 18. Then for all $\varepsilon_0 > 0$, one can compute $b_0 \leq a_1 \leq b_1 \leq \cdots \leq a_m$ such that for all $j \leq m$, f does not change sign on $[b_j, a_{j+1})$, and $|f(t)| \leq \epsilon_1 = 5\varepsilon_0 \,\mathrm{MaxValue}$ over all $t \in [a_j, b_j]$ and $|b_j - a_j| \leq \varepsilon_0$.*



The proof of Lemma 20 can be found in [13]. It is crucial that the number of intervals m does not depend upon the bounds ε_0 to obtain an effectively computable mode policy. From a value function Val, associated function f, and $b_0 \leq a_1 \leq b_1 \leq \cdots \leq a_m$ from Lemma 20 using a small enough $\varepsilon_0 > 0$, we define the approximated mode policy $\overline{\sigma}$ as follows:

- Between (b_i, b_{i+1}), the policy $\overline{\sigma}$ plays α if the sign of $f(\frac{b_i+a_{i+1}}{2})$ is > 0. Indeed, between (b_i, a_{i+1}), we know that playing α brings a higher Val_{i+1} than playing β. Between (a_{i+1}, b_{i+1}), the difference is small so it does not matter much the choice α or β.
- Otherwise, $\overline{\sigma}$ plays β.

We denote $\overline{\mathrm{Val}}(\ell, v) = \mathcal{R}(\overline{\sigma}(\ell, v)) + \gamma \mathrm{IVal}(\ell', v_{r\leftarrow 0})$ the value function associated with $\overline{\sigma}$, for ℓ' reached by playing $\overline{\sigma}$ resetting r. Let Val_{+1} be the value function we would obtain by applying one iteration of Algorithm 1 from Val. As corollary of Lemma 20, we obtain:

▶ **Corollary 21.** $|\overline{Val}(\ell, x) - Val_{+1}(\ell, x)| < \epsilon_1 = 5\varepsilon_0 \,\mathrm{MaxValue}$.

We inductively compute $\overline{\mathrm{Val}}_i, \overline{\sigma}_i$ from the previously computed $\overline{\mathrm{Val}}_{i-1}, \overline{\sigma}_{i-1}$.

The proof of Theorem 8, provided in Section 5.6, shows that we can limit the effect of those successive approximations on the value of the resulting policy. We obtain $\overline{\mathrm{Val}}_i > \mathrm{Val}_i - \frac{\varepsilon}{2}$ by applying Corollary 21 repetitively for ε_0 chosen small enough. We conclude by combining this error for the i provided by Proposition 19, and using the fact that $\mathrm{Val}(\overline{\sigma}_i) \geq \overline{\mathrm{Val}}_i$.

5.3 Running example: methodology and numerical evaluation

We now roll out our procedure on an example, and compare with discretization [9].

Let us consider the DSTA $\mathcal{A}$ from Figure 1. For convenience, we copy the picture hereafter. At initialization, there is a unique mode $[0, \infty)$ in all control state, choosing action a.

After one step, only ℓ_2 has a non-null value. The value of ℓ_2 is $\mathrm{Val}_1(\ell_2, x) = \mathrm{Val}^*(\ell_2, x) = 1$ if $x > 1$ and 0 if $x < 1$, as the only transition with non-null reward is the one from ℓ_2 to . The value remains unchanged by later iterations. As there is no choice of action (only a available), ℓ_2 has a unique mode choosing a. It is not always possible to compute a closed-form solution for the value, and often one needs to keep the integral expression.

After two steps, the value in ℓ_1 is updated as $\mathrm{Val}_1(\ell_2, x) > 0$ for some x. The value is $\mathrm{Val}_2(\ell_1, x) = \mathrm{Val}_3(\ell_1, x) = 0$ for $x > 1$ and $\frac{\gamma}{2-x} \int_{t\in[x,1)} 1dt = \gamma\frac{1-x}{2-x}$ for $x < 1$, where again we can give a closed form solution. This will be updated at later steps (≥ 4) after ℓ_0 gets a non-zero value and transition from ℓ_1 to ℓ_0 can bring some contribution to the reward as well. As there is no choice of action in ℓ_1, there is a unique mode, choosing action a.

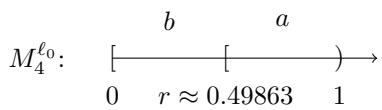
After three steps, the values of actions in ℓ_0 are updated for the first time. We have $\mathrm{Val}_3(\ell_0, x, b) = 0$ and $\mathrm{Val}_3(\ell_0, x, a) = \frac{\gamma}{2-x} \int_{t\in[x,1)} \mathrm{Val}_2(\ell_1, t)dt = \frac{\gamma^2}{2-x} \int_{t\in[x,1)} \frac{1-t}{2-t}dt > 0$. We can notice that $\frac{1-t}{2-t} = 1 - \frac{1}{2-t}$. As $\int_{t\in[x,1)} \frac{1}{2-t}dt = \int_{t\in(1,2-x]} \frac{1}{t}dt = log(2-x)$, this gives $\mathrm{Val}_3(\ell_0, x, a) = \gamma^2 \frac{1-x-log(2-x)}{2-x}$. So there is a unique mode, $[0, \infty)$ choosing action a.

The fourth steps for ℓ_0 is the first interesting case. To take its decision, value iteration considers $\mathrm{Val}_3(\ell_0, x) = \gamma^2 \frac{1-x-log(2-x)}{2-x}$ and $\mathrm{Val}_3(\ell_1, x) = \gamma \frac{1-x}{2-x}$. We have:

$$\mathrm{Val}_4(\ell_0, x, b) = \gamma^3 \int\limits_{t\in[0,1)} \frac{1-t-log(2-t)}{2-t} dt; \quad \mathrm{Val}_4(\ell_0, x, a) = \frac{\gamma^2}{1-x} \int\limits_{t\in[x,1)} \frac{1-t}{2-t} dt.$$

To know whether one needs to choose action a or b in ℓ_0 at the 4th step of value iteration, one has to study the sign of $f = \mathrm{Val}_4(\ell_0, x, b) - \mathrm{Val}_4(\ell_0, x, a)$. When it is positive, the policy should play b, and a otherwise. The value $\mathrm{Val}_4(\ell_0, x, b)$ is a constant C_1 not depending upon x, whose value can be approximated to $0.0666\gamma^3$. We *do not compute* a closed form solution for $\mathrm{Val}_4(\ell_0, x, a)$ - this is not necessary and often not possible anyway.

We have $f(x) = C_1 - \frac{\gamma^2}{1-x} \int_{t\in[x,1)} \frac{1-t}{2-t} dt$. We multiply f by $(1-x)$ which does not change its sign (as $x < 1$), defining $\tilde{f}^{(0)}(x) = f(x)(1-x)$. We compute $\frac{d\tilde{f}^{(0)}(x)}{dx} = \gamma^2 \frac{1-x}{2-x} - C_1$, getting rid of the integral. Multiplying by $(2-x)$, which is always positive, we define $\tilde{f}^{(1)}(x) = \gamma^2(1-x) - C_1(2-x)$. This is an affine function, which changes sign at most once, and whose root can be easily decided, depending on the fix value of γ. The function $\tilde{f}^{(1)}$ cancels at the same root, hence $\tilde{f}^{(0)}$, changes at most twice its sign, that is $f(x)$ changes at most twice its sign (3 modes maximum) at the same (at most 2) roots. Actually, for $\gamma = 0.95$, it has only 1 root r in [0,1), around 0.49863, so 2 modes:



We finish the example by explaining how modes are taken into account inductively:

$$\mathrm{Val}_5(\ell_0, x, b) = \gamma \int\limits_{t\in(0,r)} Val_4(\ell_0, t, b) dt + \gamma \int\limits_{t\in(r,1)} Val_4(\ell_0, t, a) dt,$$

while $\mathrm{Val}_5(\ell_0, x, a) = \mathrm{Val}_4(\ell_0, x, a)$ as $\mathrm{Val}_4(\ell_1, x, a) = 0$ for $x > 1$. By definition, $\mathrm{Val}_5(\ell_0, x, a)$ does not depend upon x, it is a constant C_2 larger than $C_1 = \mathrm{Val}_4(\ell_0, x, a)$ for $\gamma = 0.95$, which gives still 2 modes for $M_5^{\ell_0}$ with a switch at $r^5 > r$. For all i, $\mathrm{Val}_i(\ell_1, x, a)$ is a constant for $x > 1$, while $\mathrm{Val}_i(\ell_1, x, a) = Val_2(\ell_1, x, a)$ for $x < 1$. Hence, all subsequent value iteration for ℓ_0 have two modes, playing b on $(0, r^i)$ and a on $(r^i, 1)$, with r^i approaching r^* for the optimal strategy playing b on $(0, r^*)$ and a on $(r^*, 1)$.

Lemma 18 generalizes this process proving that there is only a finite number of sign changes for f, while Lemma 20 shows that the bounds of the intervals of the different roots/modes of $\tilde{f}^{(j)}$ can be approximated arbitrarily close (using Ridder's method [17]).

Evaluation on the running example and comparison with time-discretization

We now evaluate the different constants needed on the running example. Consider a standard discount $\gamma = 0.95$. MaxReward= 1 gives (proposition 19) a number of value iterations of $i < 200$ for a precision of $\varepsilon = 10^{-5}$, and of $i < 300$ for $\varepsilon = 10^{-7}$. For each iteration i, the two constants $C_1^i = Val_i(\ell_0, x, b)$ independent of the choice of $x < 1$ and $C_2^i = Val_i(\ell_1, x, a)$ independent of the choice of $x > 1$ are evaluated, i.e. 2 evaluations of function at one given point x, so overall i, 400 or 600 evaluations of function at one point. Then, the roots $root_i^1$ of $\tilde{f}^{(1)}$ are computing using C_1^i, C_2^i, then the root of $\tilde{f}^{(0)}$ in $(0, root_i^1)$ if any, and the root in $(root_i^1, 1)$ if any, using Ridder's method. This yields 600, 900 call to Ridder's altogether.

Each call needs respectively $log(2 \cdot 10^5) < 13$ and $log(2 \cdot 10^7) < 17$ iterations to ensure the $\varepsilon = \frac{1}{2}10^{-5}$ and $\varepsilon = \frac{1}{2}10^{-7}$ precision respectively. Each iteration of Ridder's method evaluates $f(x)$ at one given point x to narrow down where the root is (which side of $f(x)$ it is). Applying Corollary 21, one obtains a precision of 10^{-4} with 8200 evaluations of functions at one point, and 10^{-6} after 15900 evaluations of function at one point respectively.

Consider the time-discretization method of [9]. It cannot handle uniform distribution as used in ℓ_0, ℓ_1, but for the sake of comparison, consider a 3 states system using only exponential distributions, and a single clock. Quoting the long technical report p22 `http://www.cs.ox.ac.uk/people/taolue.chen/pub-papers/cdc_full.pdf` of [9], the authors need to solve a >60000 states MDP for 10^{-4} precision, and > 1 million states MDP for 10^{-6} precision. Using Linear Programming, this would amount to the same number of variables, with a process far from linear. Using value iteration would be possible, but incur another error which would need to be taken into account, amounting to a similar 200 iterations times 60000 states for the 10^{-4} precision. An advantage is that the time-discretization also holds for more than one clock, although the complexity would be prohibitive, with an MDP with 1.2 billion states already for 2 clocks and the precision 10^{-4}, making it a moot advantage, on top of the bounds not applying for DSTAs.

5.4 Proof of Lemma 18

Notice that if we could compute a closed-form solution of the value functions after n steps, it would be easy to deduce the values of x where the sign of $f(x)$ changes. The problem is that we do not know closed-form solutions of integrals of more and more complicated functions.

Instead, we cleverly replace the integrals by a simpler functions $\tilde{f}^{(j)}$ which changes sign at the same value, derive, until we reach a polynomial where the lemma is trivially true, and analyze the roots of function $\tilde{f}^{(j)}$ by a backward induction. This does not work with ≥ 2 clocks as for functions of ≥ 2 variables, sign changes are not related with roots of derivatives.

We fix a location ℓ at iteration i. We present the proof in the case where neither playing α nor β resets x (other cases are simpler) and all distributions on delays are uniform (which is more complex than exponentials). Other cases can be found in [13]. For $\iota \in \{\alpha, \beta\}$, we denote $(M_j = (x^\iota_j, x^\iota_{j+1}))_{0 \leq j \leq J_\iota}$ the modes of σ_i in ℓ_ι that can be reached from x. We have:

$$\mathrm{Val}_{i+1}(\ell, x, \iota) = \mathcal{R}(\iota) + \gamma \frac{1}{x^\iota_{J_\iota+1} - x} \Big(\int_{x \leq t \leq x^\iota_1} \mathrm{Val}_i(\ell_\iota, t, \sigma_i^{\ell_\iota}(M_0))dt + \sum_{j=1}^{J_\iota} \int_{x^\iota_j \leq t \leq x^\iota_{j+1}} \mathrm{Val}_i(\ell_\iota, t, \sigma_i^{\ell_\iota}(M_j))dt \Big).$$

Let $x_m = \min(x^\alpha_0, x^\beta_0)$. There exists constants C_ι, C'_ι and C''_ι such that:

$$\mathrm{Val}_{i+1}(\ell, x, \iota) = C_\iota + C'_\iota \frac{1}{x^\iota_{J_\iota+1} - x} \int_{x \leq t < x_m} \mathrm{Val}_i(\ell_\iota, t, \sigma_i^{\ell_\iota}(M_0))dt + C''_\iota \frac{1}{x^\iota_{J_\iota+1} - x} \ .$$

Lemma 18 only needs symbolic functional analysis (value of m and existence of functions $(\tilde{f}^{(j)})_{j \leq m}$), so we do not need in Lemma 18 the actual value of C_ι's, only their existence. Later, in Lemma 20, we will however need to approximate these constants. This is done through numerical integration (Newton-Cotes and Generalized Gauss–Laguerre quadrature). To study the sign of $f_{i+1}(x) = \mathrm{Val}_{i+1}(\ell, x, \alpha) - \mathrm{Val}_{i+1}(\ell, x, \beta)$ we study the sign of $\tilde{f}^{(0)} = (x^\alpha_{J_\alpha+1} - x)(x^\beta_{J_\beta+1} - x) f_{i+1}(x)$, which is the same. For C a constant, we write $\tilde{f}^{(0)}$ as:

$$\tilde{f}^{(0)}(x) = C(x^{\alpha}_{J_\alpha+1} - x)(x^{\beta}_{J_\beta+1} - x) + C'_\alpha(x^{\beta}_{J_\beta+1} - x) \int\limits_{x \le t \le x_m} \mathrm{Val}_i(\ell_\alpha, t)dt + C_\alpha(x^{\beta}_{J_\beta+1} - x)$$
$$+C'_\beta(x^{\alpha}_{J_\alpha+1} - x) \int\limits_{x \le t \le x_m} \mathrm{Val}_i(\ell_\beta, t)dt + C_\beta(x^{\alpha}_{J_\alpha+1} - x).$$

We derive $\tilde{f}^{(0)}$ once to obtain $\tilde{f}^{(1)}$, then once more obtaining for some C_1, C_2, C_3, C_4, C_5:

$$g(x) = C_1 + C_2\mathrm{Val}_i(\ell_a, x) + C_3(x^{\beta}_{J_\beta+1} - x)\mathrm{Val}'_i(\ell_\alpha, x) + C_4\mathrm{Val}_i(\ell_\beta, x) + C_5(x^{\alpha}_{J_\alpha+1} - x)\mathrm{Val}'_i(\ell_\beta, x)$$

There is no more integral terms in $g^{(2)}(x)$. Function $g^{(2)}(x)$ is of the generic form:

$$G_i(x) = P(x) + \sum_{k \le K} P^{\alpha}_k(x)\mathrm{Val}^{(k)}_i(\ell_\alpha, x) + \sum_{k \le K} P^{\beta}_k(x)\mathrm{Val}^{(k)}_i(\ell_\beta, x),$$

for some polynomials $P, P^{\alpha}_0, \ldots, P^{\alpha}_K$, $P^{\beta}_0, \ldots, P^{\beta}_K$ Generalizing the method above, we can show by induction over $i \in \mathbb{N}$ [13], that there exists a $m' \in \mathbb{N}$ such that for all $j \le m'$, there exists a sequence $(\tilde{g}^{(j)})^{m'}_{j=0}$ of derivable functions with:

- the sign of $\tilde{g}^{(0)}$ and of $G_i(x)$ coincide in all points of M,
- for $3 \le j \le m'$, the sign of $\tilde{g}^{(j)}$ and of $\frac{d\tilde{g}^{(j-1)}}{dt}$ coincide in all points of M, and
- function $\tilde{g}^{(m')}$ has a constant sign over M.

Aplying this result, it suffices to choose $m = m'+2$ and $\tilde{f}^{(j)} = \tilde{g}^{(j-2)}$ for all $2 \le j \le m'+2$.

5.5 Proof of Proposition 19

First, we define $\mathrm{MaxReward} = \max_{(\ell,\iota)\in\mathcal{L}\times E}(\mathcal{R}(\iota))$, the maximal reward over all the transitions, as well as $\mathrm{MaxValue} = \frac{\mathrm{MaxReward}}{1-\gamma}$ an upper bound on values in all the states.

We now show that the value iteration process indeed converges towards the optimal value $\mathrm{Val}^*(\mathcal{A})$. For this, we first demonstrate that $(\mathrm{Val}_i)_{i\in\mathbb{N}}$ converges as it is non-decreasing and upper bounded. Then we prove that the value it converges to is a (bounded) solution to Bellman Optimality Equations, which allows us to stop the algorithm after a computable number i of steps, with the guarantee that the policy is ε-optimal.

Proof.

Value Iteration converges. Recall that $\mathrm{Val}_0(\ell, v) = 0$ for all ℓ, v, and that all rewards are non negative. By an induction on i, we show that the Val_i have an upper bound:

$$\forall i \in \mathbb{N}, \forall(\ell, v) \in \mathcal{L} \times \mathbb{R}_{\ge 0}, \mathrm{Val}_i(\ell, v) \le \frac{\mathrm{MaxReward}}{1-\gamma} = \mathrm{MaxValue}$$

Another induction on i shows that the v_i are increasing: for all $i \ge 0$, $\mathrm{Val}_{i+1} \ge \mathrm{Val}_i$.
Initialization: $\mathrm{Val}_0 \le \mathrm{Val}_1$ is trivial as $\mathrm{Val}_0 = 0$ and no reward is negative.
Now, assume $\mathrm{Val}_{J+1} \ge \mathrm{Val}_J$. Let $(\ell, v) \in \mathcal{L} \times \mathbb{R}_{\ge 0}$.

$$\begin{array}{rcl}
\mathrm{Val}_{J+2}(\ell, v) & = & \max_{\iota\in E}\mathcal{R}(\iota) + \gamma \int_{\delta\in I(\ell', v_{[r\leftarrow 0]})} \mathrm{Val}_{J+1}(\ell', v_{[r\leftarrow 0]} + \delta)d\mu_{(\ell', v[r\leftarrow 0])} \\
& & \text{applying the induction hypothesis, we get} \\
& \ge & \max_{\iota\in E}\mathcal{R}(\iota) + \gamma \int_{\delta\in I(\ell', v_{[r\leftarrow 0]})} \mathrm{Val}_{J}(\ell', v_{[r\leftarrow 0]} + \delta)d\mu_{(\ell', v_{[r\leftarrow 0]})} \\
& = & \mathrm{Val}_{J+1}(\ell, v)
\end{array}$$

Thus (Val_i) is increasing with i, bounded, hence converges towards some Val_∞.

Val_∞ is a solution to Bellman Optimality Equation. For all $(\ell, v) \in \mathcal{L} \times \mathbb{R}_{\geq 0}$ and i:

$$\mathrm{Val}_{i+1}(\ell, v) = \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \int\limits_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \mathrm{Val}_i(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})}$$

At the limit, we obtain:

$$\begin{array}{rcl} \mathrm{Val}_\infty(\ell, v) & = & lim_{i \to \infty} \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \mathrm{Val}_i(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \\ & = & \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \lim_{i \to \infty} \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \mathrm{Val}_i(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \\ \mathrm{Val}_\infty(\ell, v) & = & \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \mathrm{Val}_\infty(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \end{array}$$

Val_∞ is indeed a bounded solution of the Bellman Optimality Equation. Under the hypothesis that the optimal value Val^* exists, it is the only bounded solution of the Bellman Optimality Equation: we therefore have $\mathrm{Val}_\infty = \mathrm{Val}^*$.

As the mode-policies produced by Value Iteration are greedy over the successive Val_i that converge toward Val^*, their values converge toward Val^*.

Expression of the approximation as a function of i. As stated, σ_i^* plays optimally according to Val_i, thus $\mathrm{Val}_{\sigma_i^*} \geq \mathrm{Val}_i$. Furthermore, by unraveling the expression of Val_i up to $\mathrm{Val}_0 = 0$ and comparing it to Val_∞ (using the Bellman Optimality Equation), we obtain: $\mathrm{Val}_\infty(\ell, v) - \mathrm{Val}_i(\ell, v) \leq \gamma^i \,\mathrm{MaxValue}$. Given $\varepsilon > 0$, we set $i = \frac{\log(\varepsilon) - \log(1-\gamma) + \log(\mathrm{MaxReward})}{\log(\gamma)}$, to obtain $\mathrm{Val}_{\sigma_i^*} \geq \mathrm{Val}_i \geq \mathrm{Val}_\infty - \varepsilon = \mathrm{Val}^* - \varepsilon$. ◀

5.6 Proof of Theorem 8

Let $\varepsilon > 0$ be a small constant. Using Proposition 19 we know that we can compute i such that $\mathrm{Val}_i^* \geq \mathrm{Val}^* - \frac{\varepsilon}{2}$. In particular, $\mathrm{Val}_i^*(\lambda) \geq \mathrm{Val}^*(\lambda) - \frac{\varepsilon}{2}$ (where λ is the empty run). We choose approximation factor ε_0 as follows for Corollary 21 such that $\overline{\mathrm{Val}}_i(\lambda) \geq \mathrm{Val}_i^*(\lambda) - \frac{\varepsilon}{2}$.

We show by induction on j that $\overline{\mathrm{Val}}_j \geq \mathrm{Val}_j^* - \sum_{k=0}^{j-1} \gamma^k \times (5\varepsilon_0 \,\mathrm{MaxValue})$ with ε_0 as defined in Lemma 20. For $j = 0$ we have $\overline{\mathrm{Val}}_0 = \mathrm{Val}_0$. Inductive step: suppose the result for a given j. Then, by Corollary 21 and Bellman Optimality Equation:

$$\overline{\mathrm{Val}}_{j+1}(\ell, v) \geq \max_\iota \left(\mathcal{R}(\iota) + \gamma.\overline{\mathrm{IVal}}_j(loc'_\iota, v_{[r_\iota \leftarrow 0]})\right) - 5\varepsilon_0 \,\mathrm{MaxValue}$$

with r_ι the reset associated with ι in $(\ell, v + \delta)$ and ℓ'_ι the target location. Using the induction hypothesis we then have

$$\overline{\mathrm{Val}}_{j+1}(\ell, v) \geq \max_\iota \left(\mathcal{R}(\iota) + \gamma.\mathrm{IVal}_j(loc'_\iota, v_{[r_\iota \leftarrow 0]})\right) - 5\,\mathrm{MaxValue}\left(\gamma \times \sum_{k=0}^{j-1} \gamma^k \varepsilon_0 - \varepsilon_0\right).$$

By definition of Val_{j+1} we then have our result which finished the proof by induction.

We thus fix $\frac{\varepsilon}{2} \geq \sum_{k=0}^{i-1} \gamma^k 5\varepsilon_0 \,\mathrm{MaxValue}$ i.e. $\varepsilon_0 \leq \frac{\varepsilon}{10\,\mathrm{MaxValue}} \times \frac{(1-\gamma)}{(1-\gamma^i)}$, which we can effectively compute. This ends the proof of Theorem 8.

5.7 Proofs of Theorem 9 : Counter example with 2 clocks

To prove Theorem 9, we exhibit a simple DSTA with two clocks and without loops (Figure 2) in which the optimal value can not be represented using polytopes.

The only choice is between e_2 and e_3 in ℓ_1. The optimal values in ℓ_4, ℓ_5 and ℓ_6 are simply 0 and the optimal values in ℓ_2 and ℓ_3 can be correctly computed in one step as: $\mathrm{Val}^*(\ell_2, (x, y)) = 1$ if $x > 1$ and 0 otherwise and $\mathrm{Val}^*(\ell_3, (x, y)) = 2$ if $y \leq 1$ and 0 otherwise.

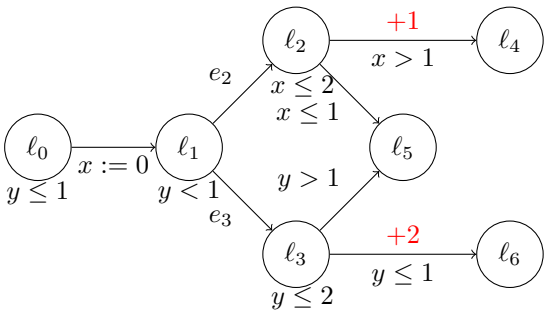


Figure 2 A simple DSTA with two clocks without loops.

Thus the values of e_2 and e_3 are correctly computed in two steps as:

$$\mathrm{Val}^*(\ell_1, (x, y), e_2) = \frac{1}{2-x} \int\limits_{\delta \in]1-x, 2-x]} 1dt = \frac{1}{2-x}$$

$$\mathrm{Val}^*(\ell_1, (x, y), e_3) = \frac{2}{2-y} \int\limits_{\delta \in]0, 1-y]} 1dt = 2 \times \frac{1-y}{2-y}$$

The optimal value in ℓ_1 is thus maximum of these two values. It comes that e_2 is played when $\frac{1}{2-x} - 2 \times \frac{1-y}{2-y} > 0$ i.e. $3y + 2x - 2xy - 2 > 0$ which cannot be represented by a polytope.

6 Conclusion

Over the span of this paper, we defined a framework for Reinforcement Planning in dense time with discontinuities, modelled by DSTAs. In the presence of a single clock, we developed an algorithm that computes a ε-optimal policy over an uncountable number of configurations in bounded time, represented in a finite way as a mode policy. Notice that we do not use a closed-form solution for the value function itself: we believe such closed-form solutions cannot be computed in general even for 1-clock DSTA, as some integrals do not have closed-form solutions. Further, we did not rely on a discretization of time in ϵ-steps, in order to keep the number of mode minimal which we showed to be efficient on our running example w.r.t. time-discretization. This made the proofs particularly involved.

An interesting question is what happens if we use policy iteration instead of value iteration. This technique considers the reward associated with a policy, and thus demands more complex computations than value iteration which only considers bounded paths. We do not know how to evaluate them as we do not have closed-form solution of the value function.

Last, we supposed that we knew all the parameters of our model (rewards, distributions on delays) as it is the case in RP. In untimed settings, Reinforcement Learning developed techniques to work on partially unknown models by exploring them while trying to limit the necessary sample size. A interesting extension of this work would be to study a possible adaptation of exploration algorithms in dense time, which however reveals to be very challenging when dealing with Timed automata and its extensions [11, 14].

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. doi:10.1007/BFb0031987.
2. Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Probabilistic and topological semantics for timed automata. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science: 27th International Conference, New Delhi, India, December 12-14, 2007. Proceedings*, pages 179–191, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-540-77050-3_15.
3. Christel Baier, Holger Hermanns, Joost-Pieter Katoen, and Boudewijn R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time markov decision processes. *Theoretical Computer Science*, 345(1):2–26, 2005. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004). doi:10.1016/j.tcs.2005.07.022.
4. Nathalie Bertrand, Thomas Brihaye, and Blaise Genest. Deciding the value 1 problem for reachability in 1-clock decision stochastic timed automata. In Gethin Norman and William Sanders, editors, *Quantitative Evaluation of Systems*, pages 313–328. Springer International Publishing, 2014. doi:10.1007/978-3-319-10696-0_25.
5. Nathalie Bertrand and Sven Schewe. Playing optimally on timed automata with random delays. In Marcin Jurdziński and Dejan Ničković, editors, *Formal Modeling and Analysis of Timed Systems*, pages 43–58, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-33365-1_5.
6. Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007.
7. Steven Bradtke and Michael Duff. Reinforcement learning methods for continuous-time markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL: https://proceedings.neurips.cc/paper_files/paper/1994/file/07871915a8107172b3b5dc15a6574ad3-Paper.pdf.
8. Tomas Brazdil, Vojtech Forejt, Jan Krcal, Jan Kretinsky, and Antonin Kucera. Continuous-Time Stochastic Games with Time-Bounded Reachability. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61–72, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FSTTCS.2009.2307.
9. Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Reachability probabilities in markovian timed automata. In *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'11)*, pages 7075–7080, 2011. doi:10.1109/CDC.2011.6160992.
10. Kenji Doya. Reinforcement Learning in Continuous Time and Space. *Neural Computation*, 12(1):219–245, January 2000. doi:10.1162/089976600300015961.
11. Olga Grinchtein. *Learning of Timed Systems*. PhD thesis, Uppsala University, Sweden, 2008. URL: http://nbn-resolving.de/urn:nbn:se:uu:diva-8763.
12. Xianping Guo and On'esimo Hernandez-Lerma. Continuous-time markov decision processes. *Theory and Applications*, 2009. doi:10.1007/978-3-642-02547-1.
13. Léo Henry, Blaise Genest, and Alexandre Drewery. Reinforcement planning for effective ε-optimal policies in dense time with discontinuities. Technical report, CNRS, 2023. URL: http://perso.crans.org/genest/HGD23.pdf.
14. Léo Henry, Thierry Jéron, and Nicolas Markey. Active learning of timed automata with unobservable resets. In Nathalie Bertrand and Nils Jansen, editors, *18th International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS'20)*, volume 12288 of *Lecture Notes in Computer Science*, pages 144–160. Springer, September 2020. doi:10.1007/978-3-030-57628-8_9.

15 Rupak Majumdar, Mahmoud Salamati, and Sadegh Soudjani. On Decidability of Time-Bounded Reachability in CTMDPs. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 133:1–133:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.133.

16 Markus N. Rabe and Sven Schewe. Optimal time-abstract schedulers for ctmdps and continuous-time markov games. *Theoretical Computer Science*, 467:53–67, 2013. doi:10.1016/j.tcs.2012.10.001.

17 C. Ridders. A new algorithm for computing a single root of a real continuous function. *IEEE Transactions on Circuits and Systems*, 26(11):979–980, 1979. doi:10.1109/TCS.1979.1084580.

18 Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

19 Çagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, 2021. URL: https://api.semanticscholar.org/CorpusID:231855323.

Parameterized Complexity of Incomplete Connected Fair Division

Harmender Gahlawat
Ben-Gurion University of the Negev, Beersheba, Israel

Meirav Zehavi
Ben-Gurion University of the Negev, Beersheba, Israel

Abstract

Fair division of resources among competing agents is a fundamental problem in computational social choice and economic game theory. It has been intensively studied on various kinds of items (*divisible* and *indivisible*) and under various notions of *fairness*. We focus on Connected Fair Division (CFD), the variant of fair division on graphs, where the *resources* are modeled as an *item graph*. Here, each agent has to be assigned a connected subgraph of the item graph, and each item has to be assigned to some agent.

We introduce a generalization of CFD, termed Incomplete CFD (ICFD), where exactly p vertices of the item graph should be assigned to the agents. This might be useful, in particular when the allocations are intended to be "economical" as well as fair. We consider four well-known notions of fairness: PROP, EF, EF1, EFX. First, we prove that EF-ICFD, EF1-ICFD, and EFX-ICFD are W[1]-hard parameterized by p plus the number of agents, even for graphs having constant *vertex cover number* (vcn). In contrast, we present a randomized FPT algorithm for PROP-ICFD parameterized only by p. Additionally, we prove both positive and negative results concerning the kernelization complexity of ICFD under all four fairness notions, parameterized by p, vcn, and the total number of different valuations in the item graph (val).

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Fair Division, Kernelization, Connected Fair Allocation, Fixed parameter tractability

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.14

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2310.01310 [26]

Funding This research was supported by the European Research Council (ERC) grant titled PARAPATH.

1 Introduction

Allocating indivisible goods among competing agents in a "fair" manner is a fundamental research problem in computational social choice [10]. Classically, this resource allocation problem is referred to as Fair Division, and it has been intensively studied in the literature under various notions of *fairness* and *efficiency* [3, 4, 7, 11, 14, 16, 17, 29].

It might be desirable, sometimes, that the allocations respect some connectivity measure. For example, while allocating offices to various research teams in a university campus, it might be desirable to provide adjoining offices to the members of the same team. Similarly, while dividing farmlands among heirs, it might be desirable to provide a connected piece to each individual. Another example can be when the items are people and the graph is a social network, and we want each team to consist of members that know each other (not necessarily directly). Thus, Bouveret et al. [9] introduced Connected Fair Division (CFD), an adaptation of Fair Division to graphs. In this problem, we are given (1) a set of items represented as vertices of an input graph G, (2) a set of agents A, and (3) a set $\mathcal{U}$ of utility

© Harmender Gahlawat and Meirav Zehavi;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 14; pp. 14:1–14:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 Summary of our results.

Fairness	$p + \mathsf{vcn} + \lvert A\rvert$	$p + \mathsf{vcn} + \lvert A\rvert + \mathsf{val}$	
	Parameterization	**Exponential Kernel**	**No-Poly Kernel**
EF	**W[1]-hard [Theorem 1]**	**[Theorem 3]**	**[Theorem 4]**
EF1	**W[1]-hard [Theorem 2]**	**[Theorem 3]**	**[Theorem 4]**
EFX	**W[1]-hard [Theorem 2]**	**[Theorem 3]**	**[Theorem 4]**
PROP	**FPT by p [Theorem 5]**	**[Theorem 3]**	**[Theorem 4]**

functions specifying the utility of each item (vertex) for each agent. The goal is to decide whether there exists an allocation of vertices to the agents which is (i) *fair* under some notion of fairness, and (ii) *connected* (i.e., for each agent, the subgraph induced by the set of vertices assigned to the agent is connected). Fixing a fairness notion φ, CFD is referred to as φ-CFD. Note that Fair Division is the special case of CFD restricted on complete graphs. Since its introduction, CFD has received a significant amount of interest [3, 6, 20, 28].

We introduce an "economic" generalization of CFD, termed Incomplete CFD (ICFD), where we have to allocate exactly p vertices of G to the agents (i.e., $|V(G)|-p$ vertices remain unassigned), and where each agent must be assigned at least one vertex. Note that CFD is the special case of ICFD when $p = |V(G)|$. Leaving some items unassigned has recently gained significant attention in context of classical Fair Division [5, 8, 15, 18]. Here, the motivation is to attain *fairness* by donating some items to *charity*. We note that the variation of ICFD that asks to allocate at most (rather than exactly) k vertices can be modeled using ICFD (by repeating for $p = 1$ to k).

Our Contribution. In this paper, we study the parameterized complexity of ICFD considering four well-known notions of fairness: PROP, EF, EF1, and EFX (defined in Section 2). An overview of our results is given in Table 1. We remark that parameterized analysis of fair allocations is an extensively studied research subject [7, 8, 20, 24, 21]. We first prove that EF-ICFD remains W[1]-hard when parameterized by $\mathsf{vcn} + p + |A|$, where vcn is the *vertex cover number* of the item graph and $|A|$ is the total number of agents. To this end, we provide a simple parameterized reduction from the (k, M)-SUM problem (parameterized by k) to EF-ICFD on a star graph parameterized by $p + |A|$. Moreover, we have $|A| = 2$, and both agents have *identical valuation* in the reduced instance.

▶ **Theorem 1.** EF-ICFD *is $W[1]$-hard parameterized by $p + |A|$ even for star graphs.*

Next, we extend this result to the fairness notions EF1 and EFX. Specifically, we show that EF1-ICFD and EFX-ICFD are W[1]-hard when parameterized by $\mathsf{vcn} + p + |A|$, even on graphs having $\mathsf{vcn} = 2$. To this end, we provide a non-trivial parameterized reduction from (k, M)-SUM problem (parameterized by k) to EF1-ICFD (resp., EFX-ICFD) parameterized by $p + |A|$ on a graph having $\mathsf{vcn} = 2$ (and $|A| = 3$).

▶ **Theorem 2.** EF1-ICFD *and* EFX-ICFD *are $W[1]$-hard parameterized by $p + |A| + \mathsf{vcn}$ even for graphs having* $\mathsf{vcn} = 2$.

Observe that φ-ICFD, for $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, is trivially in the class XP when parameterized by p alone. Since φ-ICFD, for $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, is W[1]-hard when parameterized by $p + \mathsf{vcn} + |A|$, we include another parameter in quest of tractability. Let val be the total number of distinct valuations that agents assign to items (val is the range of the function

$\mathcal{U}$). We note that val is used to achieve tractability of CFD by Deligkas et al. [20]. We establish that φ-ICFD, for $\varphi \in \{\mathsf{PROP}, \mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, is FPT parameterized by $\mathsf{vcn} + \mathsf{val} + p$ by designing exponential kernels.

▶ **Theorem 3.** *For $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, φ-ICFD admits a kernel with at most $p2^{\mathsf{vcn}}\mathsf{val}^{\mathsf{val}} + \mathsf{vcn}$ vertices. Moreover, PROP-ICFD admits a kernel with at most $p2^{\mathsf{vcn}}\mathsf{val}^{\mathsf{val}} + \mathsf{vcn} + p$ vertices.*

Next, we complement our exponential kernels by showing that it is unlikely for φ-ICFD, for $\varphi \in \{\mathsf{PROP}, \mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, parameterized by $\mathsf{vcn} + \mathsf{val} + |A| + p$ to admit a polynomial compression. To this end, we provide *polynomial parameter transformations* from Red-Blue Dominating Set.

▶ **Theorem 4.** *For $\varphi \in \{\mathsf{PROP}, \mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, φ-ICFD parameterized by $\mathsf{vcn} + \mathsf{val} + |A| + p$ does not admit polynomial compression, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Finally, we establish that PROP-ICFD is FPT when parameterized by p alone. To this end, we provide a color-coding based randomized algorithm with constant success probability[1].

▶ **Theorem 5.** *There exists a randomized algorithm that solves PROP-ICFD, in time $e^p p^{\mathcal{O}(p \log p)} m^{\mathcal{O}(1)}$ with success probability at least $1 - \frac{1}{e}$.*

The choice of vcn as a structural parameter is quite natural for our results. Arguably, vcn is the best structural parameter for providing negative results – indeed, our negative results on parameterized complexity and kernelization complexity of ICFD parameterized by vcn (plus other relevant parameters) imply the same for other smaller parameters such as treewidth, cliquewidth, treedepth, and feedback vertex set (plus other relevant parameters) [19]. Furthermore, vcn is one of the most efficiently computable parameters from both approximation [34] and parameterized [19] points of view, making it fit from an applicative perspective even when a vertex cover is not given along with the input.

Brief Survey. The need for fairly dividing resources among competing agents is one of the oldest problems in human civilization. The famous cut-and-choose method to remove envy between two agents dates back to the Book of Genesis. Fairly dividing a divisible item among agents is a very old and classical problem, also referred to as *cake-cutting*, and is studied extensively in the literature [13, 33]. When the items are indivisible, the problem is well studied under the name Fair Division [3, 4, 7, 11, 12, 14, 16, 17, 29]. The study of CFD was initiated by Bouveret et al. [9]. Later, their work was extended to include the notion of *chores* as well by Aziz et al. [3]. Various notions of fairness like maximin share allocation (Greco and Scarcello [27]) and Pareto-optimal allocations (Igarashi and Peters [28]) are also studied. Bilò et al. [6] studied the conditions that guarantee various notions of fair allocations. Recently, Deligkas et al. [20] provided a comprehensive picture of the parameterized complexity of CFD.

Motivation. A wide range of considerations motivates our definition of ICFD. First, it is often the case that the allocator wants to save some resources for later. For instance, consider the example of a university campus where the professors and research groups are first allocated offices while saving some rooms for various administrative purposes and classrooms, or for future professors in case the university is expanding. In these settings, the requirement of connectivity might be desirable.

[1] Clearly, repetition allows to boost the success probability to any constant.

Second, partial allocations are specifically useful when the allocator wants to save some resources for the future. For instance, it is practical for a university department to not allocate all of its resources (like computer equipment and travel funding) in one round of allocations and save some for future use. The notion of connectivity can be introduced based on the graph of devices that work well with each other. A similar setting arises while allocating ground to companies/people to build buildings while saving some ground for the future. Here, the connectivity requirement comes naturally.

Third, it saves resources from the viewpoint of the allocators: using this framework, they can assign the least amount of resources that makes the agents "happy". This is best exemplified when the allocator actually needs to buy the items. For example, when a company actually needs to rent offices in an office space or has to buy some equipment for its employees.

For another example, consider a park where different groups want to organize picnics on a specific day, and the park committee has to allocate these picnic spots. Allocating a connected spot to each group makes good sense here. Further, each day, the committee receives multiple applications with the preferences of each group. Now, it may be impractical to delay the allocation process till the very last day. So, it may be a good idea to allocate these spots in phases such that in each phase, the applicants are provided the spots "fairly" while saving a sufficient amount of spots for further phases.

Due to the connectivity requirement, it might so happen that there are some "problematic" vertices, which, when assigned to one of the agents, may deem the allocation "unfair" in CFD. For example, consider a vertex v such that ℓ degree-one vertices are attached to it. Moreover, let n be the number of agents, and ℓ is much larger than n. In this case, since each vertex is to be assigned to some agent in CFD, at least $\ell - n$ of these degree-one vertices must be assigned to the same agent that is assigned the vertex v, possibly making other agents "envious" of this agent. In these scenarios, it is a practical (and desirable) question to seek a fair allocation by leaving some vertices unassigned.

2 Preliminaries

For $\ell \in \mathbb{N}$, let $[\ell] = \{1, \ldots, \ell\}$ and $[0, \ell] = \{0\} \cup [\ell]$. Let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Due to space constraints, the proofs of the statements marked with $(*)$, or for which we only give a proof sketch, can be found in the full version [26].

Graph Theory. We consider finite, simple, and connected graphs. For a graph G, let $V(G)$ and $E(G)$ denote the *vertex set* and the *edge set* of G, respectively. For a vertex $v \in V(G)$, let $N(v)$ denote the *open neighborhood* of v, that is, $N(v) = \{u \mid uv \in E(G)\}$. For a subset $X \subseteq V(G)$, let $N_X(v) = N(v) \cap X$. Moreover, let $G[X]$ denote the subgraph of G induced by vertices in X, and let $G - X = G[V(G) \setminus X]$ of G. A set $U \subseteq V(G)$ is a *vertex cover* of G if for every edge in G, at least one of its endpoints is in U. The minimum cardinality of a vertex cover of G is its *vertex cover number* (vcn). Given a graph G, let $d_G(v) = |N(v)|$. For standard graph theoretic terminology not defined explicitly in this paper, we refer to the book by Diestel [22].

Incomplete Connected Fair Division. An instance of Incomplete Connected Fair Division (ICFD) consists of $(G, A, \mathcal{U}, p)$ where G is the *utility graph* on m vertices, $A = [n]$ is the set of n *agents*, $\mathcal{U}$ is the set of *utility functions* $\{u_i : V(G) \to \mathbb{N}_0\}_{i \in [n]}$, and $p \in [m]$. When clear from context, we denote $|V(G)|$ by m and $|A|$ by n. Every vertex $v \in V(G)$

corresponds to an *item*. It is a standard assumption in the literature, and we assume it too, that the agents have *additive valuations*, i.e., for every $X \subseteq V(G)$ and for every $i \in A$, we have $u_i(X) = \sum_{v \in X} u_i(v)$. An *allocation* of items is a tuple $\Pi = (\pi_1, \ldots, \pi_n)$ such that:

1. For $i \in A$: $\pi_i \subseteq V(G)$, $|\pi_i| \geq 1$, and $G[\pi_i]$ is connected.
2. For $i, j \in A$ such that $i \neq j$: $\pi_i \cap \pi_j = \emptyset$.
3. $|\bigcup_{i \in A} \pi_i| = p$.

We say that the *bundle* π_i is *assigned* to agent i in Π. For a bundle π_i, let $\tau_i = \{v \in \pi_i \mid G[\pi_i \setminus \{v\}] \text{ is connected}\}$. Next, we have the following notions of *fairness* (that we consider in this paper). An allocation $\Pi = (\pi_1, \ldots, \pi_n)$ is:

- *proportional* (PROP) if for every $i \in A$, $u_i(\pi_i) \geq \frac{u_i(V(G))}{n}$;
- *envy-free* (EF) if for every $i, j \in A$, $u_i(\pi_i) \geq u_i(\pi_j)$;
- *envy-free up to one item* (EF1) if for every $i, j \in A$, $u_i(\pi_i) \geq u_i(\pi_j) - \max_{v \in \tau_j} u_i(v)$;
- *envy-free up to any item* (EFX) if for every $i, j \in A$, $u_i(\pi_i) \geq u_i(\pi_j) - \min_{v \in \tau_j} u_i(v)$.

For a fairness criterion $\varphi \in \{\mathsf{PROP}, \mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, $\varphi-$ICFD asks whether there exists an allocation Π for $(G, A, \mathcal{U}, p)$ satisfying φ. An allocation Π that satisfies φ is termed as *φ-allocation*. Agents i and j have *same type* if for every $v \in V(G)$, $u_i(v) = u_j(v)$. Let $\mathcal{A}$ denote the set of all agent types, i.e., $\mathcal{A}$ is the partition of A such that agents i and j are in the same part $\mathsf{a} \in \mathcal{A}$ if and only if i and j have the same type. The type of an agent i is the type $\mathsf{a} \in \mathcal{A}$ such that $i \in \mathsf{a}$; let a_i denote the type of agent i. Since the agents of each type have the same valuation function, we let u_{a} denote the valuation function u_i where $i \in \mathsf{a}$. When $|\mathcal{A}| = 1$, we say that the agents have *identical valuations* and for each vertex v and agent i, we also denote $u_i(v)$ by $u(v)$ for brevity. We say that an instance $(G, A, \mathcal{U}, p)$ admits *unary valuation* (resp., *binary valuation*) if the valuations (in the valuation function $\mathcal{U}$) are encoded in unary (resp., binary), i.e., there is some polynomial (resp., exponential) function f such that $\max_{i \in A, v \in V(G)} u_i(v) \leq f(|V(G)|)$.

Parameterized Complexity. In parameterized complexity, each instance of a problem Π is associated with a non-negative integer *parameter* k. A parametrized problem Π_p is *fixed-parameter tractable* (FPT) if there is an algorithm that, given an instance (I, k) of Π_p, solves it in time $f(k) \cdot |I|^{\mathcal{O}(1)}$, for some computable function f. Central to parameterized complexity is the following hierarchy of complexity classes: $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots \subseteq \mathsf{XP}$. We note that $\mathsf{FPT} \neq \mathrm{W}[1]$ under ETH.

Two instances I and I' (possibly of different problems) are *equivalent* when I is a Yes-instance if and only if I' is a Yes-instance. A *compression* of a parameterized problem Π_1 into a (possibly non-parameterized) problem Π_2 is a polynomial-time algorithm that maps each instance (I, k) of Π_1 to an equivalent instance I' of Π_2 such that size of I' is bounded by $g(k)$, for a computable function $g(\cdot)$. If $g(\cdot)$ is polynomial, then the problem is said to admit a *polynomial compression*. When $\Pi_1 = \Pi_2$, compression is also termed as *kernelization*. A *polynomial parameter transformation* from Π_1 to Π_2 is a polynomial time algorithm that given an instance (I, k) of Π_1 generates an equivalent instance (I', k') of Π_2 such that $k' \leq p(k)$, for some polynomial $p(\cdot)$. Here, if Π_1 does not admit a polynomial compression, then Π_2 does not admit a polynomial compression [19]. We refer to the books by Cygan et al. [19] and Fomin et al. [25] for more details on parameterized complexity.

Preliminary Results and Observations. Consider an instance $(G, A, \mathcal{U}, p)$ of ICFD. For each $i \in A$, we assume that $u_i(V(G)) > 0$. Moreover, since each agent must be assigned at least one vertex, we assume that $|A| > p$. We have the following basic observations and lemmas.

▶ **Observation 6** (∗). *Let Π be an allocation for an instance $(G, A, \mathcal{U}, p)$ of* EF-ICFD *with identical valuations (i.e., $|\mathcal{A}| = 1$). Then, Π admits* EF *if and only if for every $i, j \in A$, $u(\pi_i) = u(\pi_j)$.*

▶ **Lemma 7** (∗). *Consider an instance $(G, A, \mathcal{U}, p)$ of φ-*ICFD *where $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$. Let H be an induced subgraph of G. If $(H, A, \mathcal{U}, p)$ is a Yes-instance of φ-*ICFD*, then $(G, A, \mathcal{U}, p)$ is a Yes-instance of φ-*ICFD*.*

We remark that for an induced subgraph H of G, it might happen that $(H, A, \mathcal{U}, p)$ is a Yes-instance of PROP-ICFD but $(G, A, \mathcal{U}, p)$ is a No-instance of PROP-ICFD. To see this, consider a Yes-instance $(H, A, \mathcal{U}, p)$ of PROP-ICFD such that $H = G - \{v\}$ and $|A| > 1$. Now, for each agent $i \in A$, let $u_i(v) = (n + 1) \sum_{x \in V(H)} u_i(x)$. Now, consider the instance $(G, A, \mathcal{U}, p)$. Since $|A| > 1$, there is at least one agent, say, j, such that the vertex v is not assigned to the agent j. Observe that even if $\pi_j = V(G) \setminus \{v\}$, $u_j(\pi_j) < \frac{u_j(V(G))}{n}$ (assuming $u_j(V(H)) > 0$). Hence, $(G, A, \mathcal{U}, p)$ is a No-instance. Next, we have the following observation concerning PROP-ICFD.

▶ **Observation 8** (∗). *Let $p < p' \leq |V(G)|$. Moreover, let $(G, A, \mathcal{U}, p)$ be a Yes-instance of* PROP-ICFD*. Then, $(G, A, \mathcal{U}, p')$ is a Yes-instance of* PROP-ICFD *as well.*

The following observation follows directly from the definition of EF1-ICFD and EFX-ICFD.

▶ **Observation 9.** *Consider an allocation $\Pi = (\pi_1, \ldots, \pi_n)$ for an instance $(G, A, \mathcal{U}, p)$. If there are two agents i and j such that for each vertex $v \in \tau_j$, $u_i(\pi_i) < u_i(\pi_j) - u_i(v)$, then Π is neither an* EF1 *allocation nor an* EFX *allocation.*

3 W[1]-hardness Results

In this section, we show that φ-ICFD where $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$ is W[1]-hard when parameterized by $p + \mathsf{vcn} + |A|$. For this purpose, we first define the following problem. The (k, M)-SUM problem is to determine, given N integers $a_1, \ldots, a_N \in [0, M]$ and a target integer $t \in [0, M]$, whether there exists $S \subseteq N$ such that $|S| = k$ and $\sum_{i \in S} a_i = t$. We also assume that $2 \leq k \leq N - 2$ (as otherwise, we can solve the problem in polynomial time). Abboud et al. [1] proved the following:

▶ **Proposition 10** ([1]). *(k, N^{2k})-SUM parameterized by k is $W[1]$-complete.*

3.1 EF-ICFD

First, we provide a parameterized reduction from (k, M)-SUM to EF-ICFD on a star graph such that $p = k + 2$ and $|A| = 2$. We begin by explaining our construction.

Construction. Let $(a_1, \ldots, a_N, t)$ be an instance of (k, M)-SUM. We create an instance $(G, A, \mathcal{U}, p)$ of EF-ICFD in the following manner. Let G be a star on $N + 2$ vertices such that $V(G) = \{v_1, \ldots, v_N\} \cup \{d_1, d_2\}$ and $E(G) = \{d_1 d_2\} \cup \{d_1 v_i \mid i \in [N]\}$ (i.e., G is a star with $N + 1$ leaves and d_1 as the center vertex). Let $A = \{1, 2\}$. Moreover, for $j \in A$ and $i \in [N]$, let $u_j(v_i) = a_i$, $u_j(d_1) = 1 + \sum a_i$, and $u_j(d_2) = u_j(d_1) + t$ (here, $|\mathcal{A}| = 1$). Finally, we set $p = k + 2$.

Now, we have the following lemma.

▶ **Lemma 11.** *$(a_1, \ldots, a_N, t)$ is a Yes-instance of (k, M)-SUM if and only if $(G, A, \mathcal{U}, p)$ is a Yes-instance of* EF-ICFD*.*

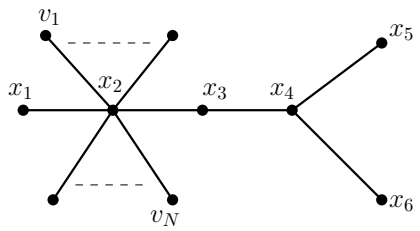


Figure 1 Illustration for the construction of graph G used in the proof of Theorem 2.

Proof Sketch. In one direction, let $S \subseteq [N]$ is a set such that $|S| = k$ and $\sum_{i \in S} a_i = t$. Then, observe that the allocation $\Pi = (\pi_1, \pi_2)$ as $\pi_1 = \{d_1\} \cup \{v_i \mid i \in S\}$ and $\pi_2 = d_2$ satisfies EF.

In the other direction, suppose $(G, A, \mathcal{U}, p)$ is a Yes-instance of EF-ICFD and $\Pi = (\pi_1, \pi_2)$ is an allocation for $(G, A, \mathcal{U}, p)$ satisfying EF. Since G is a star graph and $p \geq 4$ (as $p = k+2$ and $k \geq 2$), at least one of the agents, wlog let π_1, is assigned at least two vertices, one of which is vertex d_1. Now, $u(\pi_1) \geq 1 + \sum_{i \in [N]} u(v_i)$. Observe that the vertex $d_2 \in \pi_2$, as otherwise $u(\pi_2) \leq \sum_{i \in [N]} u(v_i) < u(\pi_1)$, contradicting that Π satisfies EF (due to Observation 6). Since $d_1 \in \pi_1$, no other vertex can be assigned to π_2 while keeping $G[\pi_2]$ connected. Hence, $\pi_2 = \{d_2\}$ and $u(\pi_2) = u(d_1) + t$. Since $|\pi_2| = 1$, note that $|\pi_1| = p - 1 = k + 1$. Hence, exactly k vertices are assigned to π_i other than d_1. Moreover, since Π satisfies EF, due to Observation 6, we have $u(\pi_1) = u(\pi_2)$. Therefore, there is a set $S \subseteq [N]$ such that $|S| = k$ and $\sum_{i \in S} u(v_i) = u(\pi_2) - u(d_1) = t$. This implies that $\sum_{i \in S} a_i = t$ (since $a_i = u(v_i)$). Hence $(a_1, \ldots, a_N, t)$ is a Yes-instance. ◀

We have the following theorem due to the above construction, Proposition 10 and Lemma 11.

▶ **Theorem 1.** EF-ICFD *is $W[1]$-hard parameterized by $p + |A|$ even for star graphs.*

3.2 EF1-ICFD and EFX-ICFD

Next, we prove that EF1-ICFD and EFX-ICFD are W[1]-hard when parameterized by $p + \mathsf{vcn} + |A|$. To this end, we provide a parameterized reduction from (k, M)-SUM to EF1-ICFD (resp.,EFX-ICFD) on a graph with $\mathsf{vcn} = 2$, while $|A| = 3$, and $p = k + 6$. First, we explain our construction.

Construction. Let $(a_1, \ldots, a_N, t)$ be an instance of (k, M)-SUM. First, we will define how to construct the graph G as it is the same for both EF1-ICFD and EFX-ICFD. Consider a path on five vertices $x_1, \ldots, x_5$ and attach a vertex x_6 to x_4. Now, we add N vertices $v_1, \ldots, v_N$ and attach them to x_2 by an edge. More formally, let $V(G) = \{x_1, \ldots, x_6, v_1, \ldots, v_N\}$ and $E(G) = \{x_2 v_i \mid i \in [N]\} \cup \{x_4 x_6\} \cup \{x_i x_{i+1} \mid 1 \leq i \leq 4\}$. See Figure 1 for a reference. Next, we have $A = [3]$ and $p = k + 6$. Finally, we define the valuation functions $\mathcal{U}$ and $\mathcal{U}'$ for EF1-ICFD and EFX-ICFD, respectively. These functions will be identical except for the valuation $u_3(x_1)$ and $u'_3(x_1)$. Moreover, let $C = \sum_{i \in [N]} a_i$. The valuation functions are defined as follows:

- $u_i(v_j) = C + a_i$ for $i \in [3]$ and $j \in [N]$,
- $u_i(x_2) = 2NC$ for $i \in [3]$,
- $u_1(x_3) = u_2(x_3) = 0$ and $u_3(x_3) = 2NC + kC + t$,
- $u_1(x_4) = u_2(x_4) = 3NC + kC + t$ and $u_3(x_4) = 0$,

- $u_i(x_5) = 0$ for $i \in [3]$,
- $u_i(x_6) = 0$ for $i \in [3]$,
- $u_i(x_1) = NC$ for $i \in [3]$. Finally, $u'_3(x_1) = 0$.

First, we have the following straightforward lemma that proves one side of our reduction.

▶ **Lemma 12.** *If $(a_1, \ldots, a_N, t)$ is a Yes-instance of (k, M)-SUM, then $(G, A, \mathcal{U}, p)$ and $(G, A, \mathcal{U}', p)$ are Yes-instances of* EF1-ICFD *and* EFX-ICFD*, respectively.*

Proof Sketch. Let $S \subseteq \{a_1, \ldots a_N\}$ be a set such that $\sum_{a \in S} a = t$ and $|S| = k$. Then, consider the allocation $\Pi = (\pi_1, \pi_2, \pi_3)$ such that $\pi_1 = S \cup \{x_1, x_2\}$, $\pi_2 = \{x_4, x_5, x_6\}$, and $\pi_3 = \{x_3\}$. It is easy to see that $|\pi_1| + |\pi_2| + |\pi_3| = |S| + 6 = k + 6 = p$, and to verify that Π is an EF1-ICFD and EFX-ICFD for $(G, A, \mathcal{U}, p)$ and $(G, A, \mathcal{U}', p)$, respectively. ◀

Some Useful Observations. Next, we prove some observations that will be useful for proving the other direction of our reduction. For the observations where we do not use any of $u_3(x_1)$ and $u'_3(x_1)$, we will use only the valuation functions $\mathcal{U}$ to ease the presentation. Moreover, we remark that these observations are valid for both EF1-ICFD (assuming valuations $\mathcal{U}$) and EFX-ICFD (assuming valuations $\mathcal{U}'$). First, observe that agents 1 and 2 have identical valuations, $\{x_2, x_4\}$ is a vertex cover of G and hence $G[V(G) \setminus \{x_2, x_4\}]$ is an independent set.. Moreover, since $k \geq 2$, note that $p \geq 8$.

For the rest of this section, let $\Pi = (\pi_1, \pi_2, \pi_3)$ be an allocation that is an EF1-ICFD for $(G, A, \mathcal{U}, p)$ (resp., EFX-ICFD for $(G, A, \mathcal{U}', p)$). First, we have the following easy observation.

▶ **Observation 13** $(*)$**.** *If $x_2 \in \pi_i$, for $i \in [3]$, then $x_2 \notin \tau_i$.*

Next, we have the following observation.

▶ **Observation 14** $(*)$**.** *$x_2 \in \pi_1 \cup \pi_2$.*

Next, we have the following crucial observation.

▶ **Observation 15** $(*)$**.** *$\pi_3 = \{x_3\}$ and $\pi_2 = \{x_4, x_5, x_6\}$.*

Finally, we have our main lemma, which implies the other side of the reduction.

▶ **Lemma 16** $(*)$**.** *If $(G, A, \mathcal{U}, p)$ (resp., $(G, A, \mathcal{U}', p)$) is a Yes-instance of* EF1-ICFD *(resp.,* EFX-ICFD*), then $(a_1, \ldots, a_N, t)$ is a Yes-instance of (k, M)-SUM.*

The following theorem is a consequence of our construction, Proposition 10, and Lemmas 12 and 16.

▶ **Theorem 2.** EF1-ICFD *and* EFX-ICFD *are $W[1]$-hard parameterized by $p + |A| +$* vcn *even for graphs having* vcn $= 2$.

Theorems 1 and 2 establish that, for $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, φ-ICFD is W[1]-hard parameterized by $p + |A| + t$ when the valuations are encoded in binary. We remark that these results cannot be directly generalized to unary valuations as (k, M)-SUM is in P when M is encoded in unary.

4 Kernelization by vcn $+ p +$ val

Let $(G, A, \mathcal{U}, p)$ be an instance of ICFD. We consider the parameterization of ICFD parameterized by $\mathsf{vcn} + p + \mathsf{val}$. Recall that vcn is the vertex cover of the input graph G, $\mathcal{A}$ is the set of all agent types, and $\mathsf{val} = |\{u_i(v) \mid i \in [n], v \in V(G)\}|$. Moreover, let $\mathsf{VAL} = \{u_i(v) \mid i \in [n], v \in V(G)\}$.

Let U be a vertex cover of size t. If no such vertex cover is given, then we can compute a vertex cover U of size $t \leq 2\mathsf{vcn}$ using a polynomial-time approximation algorithm [34]. Let I be the independent set $V(G) \setminus U$. Our kernelization algorithm uses techniques similar to the ones used by Deligkas et al. [20] to provide an FPT algorithm for CFD parameterized by $\mathsf{vcn} + |\mathcal{A}| + \mathsf{val}$. We partition the vertices of I into *equivalence classes* such that for any two vertices u and v of the same equivalence class, $N_U(u) = N_U(v)$ and $u_i(v) = u_i(u)$ for each $i \in A$. Since vertices of each equivalence class are "indistinguishable" for the agents, we keep only at most p vertices from each equivalence class. We then establish that we can have at most $2^t\mathsf{val}^{|\mathcal{A}|}$ many equivalence classes, thus giving us a kernel with at most $2^t\mathsf{val}^{|\mathcal{A}|}p + t$ vertices. Below, we discuss these ideas formally.

Kernel for EF-ICFD, EF1-ICFD, and EFX-ICFD. We have the following reduction rule.

▶ **Reduction Rule 1** (RR1). *Let $(G, A, \mathcal{U}, p)$ be an instance of φ-ICFD where $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$. Let $S \subseteq I$ be a set of vertices such that $|S| > p$ and for any two vertices $u, v \in S$, $N_U(u) = N_U(v)$ and for each agent $i \in A$, $u_i(u) = u_i(v)$. Moreover, let $v_1, \ldots v_{|S|}$ be an ordering of vertices in S. Then, let $H \Leftarrow G - \{v_{p+1}, \ldots, v_{|S|}\}$.*

We have the following lemmas to prove RR1 is safe and to bound the size of our kernel.

▶ **Lemma 17** ($*$). *RR1 is safe.*

▶ **Lemma 18** ($*$). *Let $(G, A, \mathcal{U}, p)$ be an instance of φ-ICFD such that RR1 cannot be applied to $(G, A, \mathcal{U}, p)$. Then, $|V(G)| \leq 2^t \cdot val^{|\mathcal{A}|} \cdot p + t$.*

Since $|\mathcal{A}| \leq \mathsf{val}$, RR1, along with Lemma 17 and Lemma 18, imply that for $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, φ-ICFD admits a kernel with at most $p2^{\mathsf{vcn}}\mathsf{val}^{\mathsf{val}} + \mathsf{vcn}$ vertices. This gives us the desired kernel for φ-ICFD, for $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, from Theorem 3.

Kernel for PROP-ICFD. Next, we design a kernel for PROP-ICFD. First, we give an overview of the ideas leading to this kernel. As remarked in Section 2, since it might so happen that PROP-ICFD exists for a subgraph H of G but not for G, we cannot simply delete vertices as before (which might decrease the overall valuation of the graph for some agents) to get our kernel. So, we have to keep track of the "valuation that is lost" for each agent while deleting the vertices. In this quest, we augment the graph with some dummy vertices and assign them the valuation for each deleted vertex. Hence, even if we start with a unary (resp., binary) valuation instance, we might end up with an instance that does not admit unary (resp., binary) valuations. We tackle this problem by establishing that if the valuation for a dummy vertex gets "too large", then we are dealing with a No-instance. This ensures that we have a kernel that respects the unary (resp., binary) valuation if the original instance respects unary (resp., binary) valuation. Now, we discuss the details of our kernelization algorithm.

Let $(G, A, \mathcal{U}, p)$ be an instance of PROP-ICFD. First, we use the following preprocessing step to generate an augmented instance $(G', A', \mathcal{U}', p')$ in the following manner.

Preprocessing Step: We get $(G', A', \mathcal{U}', p')$ by adding n dummy agents, n dummy vertices, and setting $p' = p + n$. More formally:

1. Let $V(G') = V(G) \cup \{d_1, \ldots, d_n\}$. Fix a vertex $w \in U$. Now, let $E(G) = E(G) \cup \{d_i d_j \mid i, j \in A\} \cup \{d_1 w\}$. Moreover, let $A' = A \cup \{n+1, \ldots, 2n\}$ and $p' = p + n$.
2. For $i \in [n]$ and $v \in V(G)$, let $u'_i(v) = 2u_i(v)$. Moreover, for $i \in [n]$ and $v \in \{d_1, \ldots, d_n\}$, let $u'_i(v) = 0$ and $u'_{n+i}(d_i) = 1$. Furthermore, for $n < i \leq 2n$ and $v \notin \{d_1, \ldots, d_n\}$, let $u'_i(v) = 0$.

Now, we have the following lemma.

▶ **Lemma 19** (∗). *$(G, A, \mathcal{U}, p)$ is a Yes-instance of* PROP-ICFD *if and only if $(G', A', \mathcal{U}', p')$ is a Yes-instance of* PROP-ICFD.

Let $D = \{d_1, \ldots, d_n\}$ be the set of dummy vertices added in the Preprocessing Step. Now, we have the following reduction rule.

▶ **Reduction Rule 2** (RR2). *Let $(G', A', \mathcal{U}', p')$ be an instance of* PROP-ICFD *(obtained from an initial instance $(G, A, \mathcal{U}, p)$ after applying Preprocessing Step and possibly RR2). Let $S \subseteq I$ be a set of vertices such that $|S| > p$ and for any two vertices $u, v \in S$, $N_U(u) = N_U(v)$ and for each agent $i \in A$, $u'_i(u) = u'_i(v)$. Moreover, let $v_1, \ldots v_{|S|}$ be an ordering of vertices in S. Then, let $G'' \Leftarrow G' - \{v_{p+1}, \ldots, v_{|S|}\}$. Moreover, we modify $\mathcal{U}'$ to get $\mathcal{U}''$ in the following manner. For $i \in A'$ and $v \in V(G'')$, $u''_i(v) \Leftarrow u'_i(v)$ and, finally, for $i \in A$ ($i \in [n]$), $u''_i(d_i) \Leftarrow u''_i(d_i) + \sum_{p+1 \leq j \leq |S|} u'_i(v_j)$.*

We have the following lemma to prove that RR2 is safe.

▶ **Lemma 20** (∗). *RR2 is safe.*

Next, we have the following rule that we apply after we have applied RR2 exhaustively. This rule ensures that the valuation on any vertex does not get "much larger" than the size of the kernel.

▶ **Reduction Rule 3** (RR 3). *Let $(G', A', \mathcal{U}', p')$ be an instance we get after applying Preprocessing Step and then an exhaustive application of RR2. If for some $i \in [n]$, $u_i(d_i) > u_i(V(G') \setminus D)$, then report a No-instance.*

▶ **Lemma 21** (∗). *RR3 is safe.*

Next, we establish a bound on the size of the reduced instance. We have the following lemma.

▶ **Lemma 22** (∗). *Let $(G, A, \mathcal{U}, p)$ be an instance of* PROP-ICFD *such that G has a vertex cover U of size t. Let $(G', A', \mathcal{U}', p')$ be the instance we get after applying the Preprocessing Step and an exhaustive application of RR2. Then, $|V(G')| \leq 2^t val^{|\mathcal{A}|} p + t + n$ and $p' \leq p + n$.*

Since $|A| = n \leq p$ and $|\mathcal{A}| \leq val$, RR2, along with Lemmas 19, 20, and 22, imply that PROP-ICFD admits a $2^t val^{val} p + t + p$ vertex kernel. Moreover, due to RR3 and Lemma 21, if the initial instance respects unary (resp., binary) valuation, then the kernel also respects unary (resp., binary) valuation. Finally, the kernelization algorithms presented in this section imply the following theorem.

▶ **Theorem 3.** *For $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, φ-ICFD admits a kernel with at most $p 2^{\mathsf{vcn}} \mathsf{val}^{\mathsf{val}} + \mathsf{vcn}$ vertices. Moreover,* PROP-ICFD *admits a kernel with at most $p 2^{\mathsf{vcn}} \mathsf{val}^{\mathsf{val}} + \mathsf{vcn} + p$ vertices.*

5 Incompressibility

In this section, we complement our exponential kernels by showing that it is unlikely that φ-ICFD admits a polynomial kernel parameterized by $\mathsf{vcn} + val + p + |A|$, where $\varphi \in \{\mathsf{PROP}, \mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$. We first define the following problem. In Red-Blue Dominating Set, we are given a bipartite graph G with a vertex bipartition $V(G) = T \cup N$ and a non-negative integer k. A set of vertices $N' \subseteq N$ is said to be an *RBDS* if each vertex in T has a neighbour in N'. The aim of Red-Blue Dominating Set is to decide whether there exists an *RBDS* of size at most k in G. We assume that $k < |N|$ as otherwise, the problem becomes trivial. Dom et al. [23] proved the following:

▶ **Proposition 23** ([23]). *Red-Blue Dominating Set parameterized by $|T| + k$ does not admit a polynomial compression, unless NP $\subseteq$ coNP/poly.*

5.1 Incompressibility for PROP and EF

In this section, we provide a polynomial parameter transformation from Red-Blue Dominating Set parameterized by $|T| + k$ to PROP-ICFD and EF-ICFD parameterized by $\mathsf{vcn} + val + p + |A|$.

Polynomial Parameter Transformation. Suppose $((G, k), |T| + k)$ is an instance of Red-Blue Dominating Set parameterized by $|T| + k$ such that $V(G) = T \cup N$. For brevity, let P denote the value of the parameter $\mathsf{vcn} + \mathsf{val} + p + |A|$. First we construct a graph G' with $V(G') = T' \cup N'$ from G by adding two dummy vertices d_1 and d_2 such that $T' = T \cup \{d_1, d_2\}$, $N' = N$, and $E(G') = E(G) \cup \{d_1x, d_2x \mid x \in N'\}$. Now, we construct the instance $((G', A, \mathcal{U}, p), P)$ of ICFD such that $((G, k), |T| + k)$ is a Yes-instance if and only if $((G', A, \mathcal{U}, p), P)$ is a Yes-instance and $|P| \leq h(|T| + k)$ where h is a polynomial function. Let $p = |T| + k + 2$ and $A = \{1, 2\}$. We will define the valuation function $\mathcal{U}$ based on the fairness criterion.

Incompressiblity of PROP-ICFD. For each vertex $v \in N'$, let $u_1(v) = u_2(v) = 0$. For each vertex $v \in T' \setminus \{d_1\}$, let $u_1(v) = u_2(v) = 1$ and $u_1(d_1) = u_2(d_1) = |T| + 1$. Note that $\mathsf{val} = 3$ and $|\mathcal{A}| = 1$ (i.e., identical valuation). Observe that our reduction is a polynomial parameter transformation since $P = \mathsf{vcn} + \mathsf{val} + p + |A| \leq |T'| + 3 + |T| + k + 2 + 2 = 2|T| + k + 9$ (as T' is a vertex cover of G' and $|T'| = |T| + 2$). Hence, the following lemma implies the incompressibility of PROP-ICFD.

▶ **Lemma 24.** *$((G', A, \mathcal{U}, p), P)$ is a Yes-instance of* PROP-ICFD *if and only if $((G, k), t + k)$ is a Yes-instance of Red-Blue Dominating Set.*

Proof Sketch. In one direction, let G has an *RBDS* S of size k. Then, observe that $\Pi = (\pi_1, \pi_2)$ such that $\pi_1 = \{d_1\}$ and $\pi_2 = (T' \setminus \{d_1\}) \cup S$ is a PROP allocation for $((G', A, \mathcal{U}, p), P)$.

In the reverse direction, let $((G', A, \mathcal{U}, p), P)$ be a Yes-instance of PROP-ICFD. Then, since $u_1(V(G')) = u_2(V(G')) = 2|T| + 2$ and both agents have identical valuation, for PROP-ICFD, $u_i(\pi_1) = u_i(\pi_2) = |T| + 1$. The only way to achieve this is to assign the vertex d_1 (possibly along with some vertices from N') to one of the agents and the vertices in $T' \setminus \{d_1\}$ (along with some vertices from N') to the other agent. WLOG, assume that $d_1 \in \pi_1$ and $T' \setminus \{d_1\} \subseteq \pi_2$. Observe that $|\pi_1| \geq 1$. Therefore, $|\pi_2| \leq p - 1 = |T| + k + 1$. Since $T' \setminus \{d_1\} \subseteq \pi_2$, $|\pi_2 \cap N'| \leq k$. Moreover, since $G'[\pi_2]$ is connected, observe that each vertex in $T' \setminus \{d_1\}$ shares an edge with at least one vertex in $\pi_2 \cap N'$. Hence, $\pi_2 \cap N'$ is an *RBDS* of G containing at most k vertices. ◀

Incompressibility of EF-ICFD. Here, we distinguish two cases depending on whether $|T| + k + 2$ is odd or even. The two cases have similar proofs and hence we consider only the case when $|T| + k + 2$ is odd, and the case when $|T| + k + 2$ is even is considered in the full version [26].

Case 1: $|T| + k + 2$ is odd. In this case, for each vertex $v \in N'$, let $u_1(v) = u_2(v) = 5|T|$. For each vertex $v \in T' \setminus \{d_1\}$, let $u_1(v) = u_2(v) = 1$ and $u_1(d_1) = u_2(d_1) = 5k|T| + |T| + 1$. Note that $\mathsf{val} = 3$ and $|\mathcal{A}| = 1$ (i.e., identical valuation). Moreover, similarly to the above case for PROP-ICFD, here also $P \leq 2|T| + k + 9$. Now, we have the following lemma.

▶ **Lemma 25.** *$((G', A, \mathcal{U}, p), P)$ is a Yes-instance of* EF-ICFD *if and only if $((G, k), |T| + k)$ is a Yes-instance of* Red-Blue Dominating Set.

Proof. In one direction, let S be an *RBDS* in G with k vertices. Then,observe that $\Pi = (\pi_1, \pi_2)$ such that $\pi_1 = \{d_1\}$ and $\pi_2 = (T' \setminus \{d_1\}) \cup S$ is an EF allocation for $((G', A, \mathcal{U}, p), P)$.

In the reverse direction, let $\Pi = (\pi_1, \pi_2)$ be an EF allocation for $((G', A, \mathcal{U}, p), P)$. First, we show that $d_1 \in \pi_1 \cup \pi_2$. Targeting contradiction, assume that $d_1 \notin \pi_1 \cup \pi_2$. Since $|\pi_1| + |\pi_2| = |T| + k + 2$ is an odd number, $|\pi_1| \neq |\pi_2|$. As the valuations are identical, we have $u(\pi_1) = u(\pi_2)$ (Observation 6). Let $|\pi_1 \cap N'| = \alpha$ and $|\pi_2 \cap N'| = \beta$. Then, $u(\pi_1) = u(\pi_1 \cap N') + u(\pi_1 \cap T') = 5\alpha|T| + (|\pi_1| - \alpha) \leq 5\alpha|T| + |T| + 1$. Similarly, $u(\pi_2) \leq 5\beta|T| + |T| + 1$. We have the following two cases:

1. $\alpha \neq \beta$: Without loss of generality assume that $\alpha > \beta$. Then, $u(\pi_1) - u(\pi_2) \geq 5\alpha|T| - 5\beta|T| - |T| - 1 \geq 5|T| - |T| - 1 \geq 4|T| - 1 > 0$ (for $|T| > 0$). Hence $u(\pi_1) \neq u(\pi_2)$, which contradicts the fact that Π is an EF allocation (due to Observation 6).
2. $\alpha = \beta$: Since $|\pi_1| \neq |\pi_2|$ and $\alpha = \beta$, observe that $|\pi_1| - \alpha \neq |\pi_2| - \beta$. Hence, $u(\pi_1) - u(\pi_2) = 5\alpha|T| + (|\pi_1| - \alpha) - 5\beta|T| - (|\pi_2| - \beta) = (|\pi_1| - \alpha) - (|\pi_2| - \beta) \neq 0$. Hence $u(\pi_1) \neq u(\pi_2)$, which contradicts the fact that Π is an EF allocation (due to Observation 6).

Hence, $d_1 \in \pi_1 \cup \pi_2$. WLOG, assume $d_1 \in \pi_1$. Thus, $u(\pi_1) \geq 5k|T| + |T| + 1$. Moreover, let π_1 be assigned α vertices from N' and β vertices from $T' \setminus \{d_1\}$. (We will argue that $\alpha = \beta = 0$.) Then, $u(\pi_1) = 5k|T| + |T| + 1 + 5\alpha|T| + \beta = 5(k+\alpha)|T| + |T| + \beta + 1$. Now, let π_2 is assigned α' vertices from N' and β' vertices from $T' \setminus \{d_1\}$. Therefore, $u(\pi_2) = 5\alpha'|T| + \beta'$ where $\beta' \leq |T| + 1$. We again have the following two cases depending on whether α' equals $k + \alpha$ or not.

1. $\alpha' = k + \alpha$. In this case, note that $u(\pi_1) = u(\pi_2)$ if and only if $|T| + \beta + 1 = \beta'$. It is only possible if $\beta = 0$ and $\beta' = |T| + 1$ (since $\beta' \leq |T| + 1$).
2. $\alpha' \neq k + \alpha$. We will show that, in this case, $u(\pi_1) \neq u(\pi_2)$. Note that $u(\pi_1) - u(\pi_2) = 5(k+\alpha)|T| + |T| + \beta + 1 - 5\alpha'|T| - \beta' = 5(k+\alpha-\alpha')|T| + |T| + 1 + \beta - \beta'$. Since $\alpha' \neq k+\alpha$, $k+\alpha-\alpha' \neq 0$, and hence, $-1 \geq k+\alpha-\alpha' \geq 1$. Therefore, $-5|T| + |T| + 1 + \beta - \beta' \geq u(\pi_1) - u(\pi_2) \geq 5|T| + |T| + 1 + \beta - \beta'$, i.e., $-4|T| + 1 + \beta - \beta' \geq u(\pi_1) - u(\pi_2) \geq 6|T| + 1 + \beta - \beta'$. Since $0 \leq \beta \leq |T| + 1$ and $0 \leq \beta' \leq |T| + 1$, we have that $-|T| \leq 1 + \beta - \beta' \leq |T| + 2$. Therefore, $u(\pi_1) \neq u(\pi_2)$.

Hence, if $u(\pi_1) = u(\pi_2)$, then $\alpha' = \alpha + k$, $\beta = 0$, and $\beta' = |T| + 1$. Moreover, $1 + \alpha + \beta + \alpha' + \beta' = p = |T| + k + 1$. Therefore, $1 + \alpha + 0 + \alpha' + |T| + 1 = |T| + k + 1$, i.e, $\alpha + \alpha' = k$. Solving $\alpha' = \alpha + k$ and $\alpha + \alpha' = k$ gives us $\alpha' = k$ and $\alpha = 0$. Therefore, $\pi_1 = \{d_1\}$ and $\pi_2 = (T' \setminus \{d_1\}) \cup S$ such that $S \subseteq N'$ and $|S| = k$. Since $G'[\pi_2]$ is connected, each element in $T' \setminus \{d_1\}$ has a neighbour in S. Since $T \subseteq T' \setminus \{d_1\}$, S is an *RBDS* of G and $|S| = k$. ◀

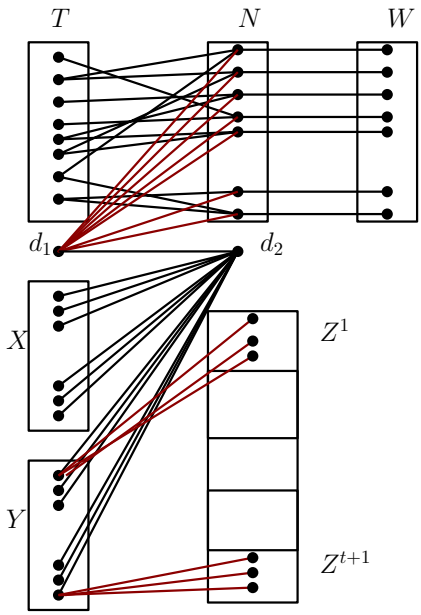


Figure 2 Illustration for the construction of G'.

5.2 Incompressibility for EFX and EF1

In this section, we provide a polynomial parameter transformation from Red-Blue Dominating Set parameterized by $|T|+k$ to EF1 (resp., EFX) parameterized by $P = \mathsf{vcn}+\mathsf{val}+|A|+p$. First, we provide our construction. Let $t = |T|$. Moreover, we assume that $k > 1$ and $t > 1$.

Construction. Suppose $((G, k), t + k)$ is an instance of Red-Blue Dominating Set parameterized by $t + k$ such that $V(G) = T \cup N$. See Figure 2 for a reference. First, we construct a graph G' with $V(G') = T' \cup N'$ from G by adding some dummy vertices. First, we add two dummy vertices d_1 and d_2 such that $T' = T \cup \{d_1\}$, $N' = N \cup \{d_2\}$, and $E(G') = E(G) \cup \{d_1x \mid x \in N'\}$ (d_1d_2 is also an edge). Next, we add two sets of vertices $X = \{x_1, \ldots, x_{t+3}\}$ and $Y = \{y_1, \ldots, y_{t+1}\}$ of size $t + 3$ and $t + 1$, respectively, to T' and attach each x_i and y_j to d_2, i.e., $T' = T' \cup \{x_1, \ldots, x_{t+3}\} \cup \{y_1, \ldots, y_{t+1}\}$ and $E(G') = E(G') \cup \{x_id_2, y_jd_2 \mid i \in [t + 3], j \in [t + 1]\}$. Next, for each y_j, $j \in [t + 1]$, we add $32kt$ (here, we want a large enough number) vertices $Z^j = \{z^j_1, \ldots, z^j_{32kt}\}$ to N' and for each z^j_ℓ, $\ell \in [32kt]$, we add the edge $y_jz^j_\ell$ to $E(G')$. Finally, let $\{v_1, \ldots, v_{|N|}\}$ be an ordering of vertices of N. For each v_i, $i \in [|N|]$, we add a vertex w_i and an edge v_iw_i to G'. Let W denote the set of vertices $\{w_1, \ldots, w_{|N}\}$. This completes our construction. Note that $|V(G')| = |V(G)| + |N| + 2 + t + 3 + t + 1 + (t + 1)32kt$. Next, we set $A = [2t + 6]$ and $p = 4k(2t + 6) = 8kt + 24k$. We define the valuation function $\mathcal{U}$ below. Moreover, for ease of presentation, let us denote the set $T \cup N \cup W \cup \{d_1\}$ by B' and the set $V(G') \setminus (B' \cup \{d_2\})$ by B.

Valuation Function. We have two types of agents: Type-I and Type-II, each with cardinality $t + 3$, such that agents of the same type have identical valuations. Recall that $|A| = 2t + 6$. Hence, without loss of generality, assume that $\{1, \ldots, t + 3\}$ are Type-I agents and $\{t + 4, \ldots, 2t + 6\}$ are Type-II agents. For ease of presentation, we will use u_{I} and u_{II} to represent the valuation function for all Type-I and Type-II agents, respectively. The valuation function $\mathcal{U}$ is defined as follows:

- for $v \in T$, $u_{\mathrm{I}}(v) = u_{\mathrm{II}}(v) = 1$,
- $u_{\mathrm{I}}(d_1) = u_{\mathrm{II}}(d_1) = 1$,
- for $v \in N$, $u_{\mathrm{I}}(v) = 0$ and $u_{\mathrm{II}}(v) = 5kt$,
- for $v \in W$, $u_{\mathrm{I}}(v) = 5kt$ and $u_{\mathrm{II}}(v) = 0$,
- $u_{\mathrm{I}}(d_2) = u_{\mathrm{II}}(d_2) = 10kt$,
- for $i \in [t+3]$, $u_{\mathrm{I}}(x_i) = 0$ and $u_{\mathrm{II}}(x_i) = 5kt + t + 1$,
- for $i \in [t+1]$, $u_{\mathrm{I}}(y_i) = 5kt + t + 1$ and $u_{\mathrm{II}}(y_i) = 0$, and
- for $i \in [t+1]$, $j \in [32kt]$, $u_{\mathrm{I}}(z_j^i) = u_{\mathrm{II}}(z_j^i) = 0$.

First, we have the following straightforward observation concerning our construction and $\mathcal{U}$.

▶ **Observation 26.** *For each connected component C in $G[B]$, $u_I(C) \leq 5kt + t + 1$ and $u_{II}(C) \leq 5kt + t + 1$.*

Now, we will show that $((G', A, \mathcal{U}, p), P)$ is Yes-instance of EF1-ICFD (resp., EFX-ICFD) if and only if $((G, k), t + k)$ is a Yes-instance of Red-Blue Dominating Set. First, we prove one side of this argument in the following lemma.

▶ **Lemma 27** (∗). *If $((G, k), t + k)$ is a Yes-instance of Red-Blue Dominating Set, then $((G', A, \mathcal{U}, p), P)$ is Yes-instance of* EF1-ICFD *as well as of* EFX-ICFD.

Next, we will prove the other side of our argument. For the rest of this section, let $((G', A, \mathcal{U}, p), P)$ be Yes-instance of EF1-ICFD (resp., EFX-ICFD) and $\Pi = (\pi_1, \ldots, \pi_{2t+6})$ be an allocation for $((G', A, \mathcal{U}, p), P)$ that satisfies EF1 (resp., EFX). We will need the following observations about the properties of Π. These observations will be valid for both EF1 as well as EFX.

Important Observations. First, we have the following easy observation.

▶ **Observation 28** (∗). *Let there be an i such that for any vertex $v \in \tau_i$, $u_I(\pi_i) - u_I(v) \geq 5t+1$ (resp., $u_{II}(\pi_i) - u_{II}(v) \geq 5t + 1$), then at most $t + 1$ agents of Type-I (resp., Type-II) can be assigned vertices only from $B' = T \cup N \cup W \cup \{d_1\}$.*

Next, we have the following crucial observation concerning Π.

▶ **Observation 29** (∗). *If $d_2 \in \pi_i$ for some $i \in A$, then $d_2 \in \tau_i$.*

The following observation establishes a necessary condition for Π to satisfy EF1 (resp., EFX).

▶ **Observation 30** (∗). *If for every vertex $v \in \tau_i$, $\pi_i \setminus \{v\}$ contains at least $k + 1$ vertices from N, then Π is not an* EF1 *(resp.* EFX*) allocation.*

The following observation establishes that "too many" vertices from B' cannot be allocated to agents in Π.

▶ **Observation 31** (∗). *For $i \in A$, π_i can be assigned at most k vertices from W and at most $k + 1$ vertices from N.*

Our next observation establishes that at least one agent of Type-I is assigned a vertex from $\{y_1, \ldots, y_{t+1}\}$, say y_i, along with some vertices (at least two) of the form z_j^i.

▶ **Observation 32** (∗). *There is at least one Type-I agent, say i, such that π_i contains at least one vertex from Y, say y_j, such that $y_j \notin \tau_i$. Equivalently, $u_I(\pi_i) - \min_{v \in \tau_i} u_I(v) \geq u_I(\pi_i) - \max_{v \in \tau_i} u_I(v) \geq 5kt + t + 1$.*

Finally, we have the following lemma that proves the other side of the reduction.

▶ **Lemma 33** (∗). *If $((G', A, \mathcal{U}, p), P)$ is Yes-instance of* EF1-ICFD *(resp.,* EFX-ICFD*), then $((G, k), t + k)$ is a Yes-instance of* Red-Blue Dominating Set.

Finally, we have the following theorem as a consequence of Proposition 23, and Lemma 24, Lemma 25, Lemma 27, Lemma 33, and the fact in all our constructions in Section 5, the value of $\mathsf{vcn} + \mathsf{val} + |A| + p$ is bounded by a function polynomial in $k + |T|$.

▶ **Theorem 4.** *For $\varphi \in \{\mathsf{PROP}, \mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, φ-ICFD parameterized by $\mathsf{vcn} + \mathsf{val} + |A| + p$ does not admit polynomial compression, unless NP $\subseteq$ coNP/poly.*

6 Color Coding For PROP-ICFD

Here, we present a color-coding [2] based FPT algorithm for PROP-ICFD parameterized by p. An (n, k)-*perfect hash family* $\mathcal{F}$ is a family of functions from $[n]$ to $[k]$ such that for every set $S \subseteq [n]$ of size k, there exists a function $f \in \mathcal{F}$ that is injective on S. We shall use the following result.

▶ **Proposition 34** ([31]). *For any $n, k \geq 1$, we can construct an (n, k)-perfect hash family of size $e^k k^{\mathcal{O}(\log k)} \log n$ and in $e^k k^{\mathcal{O}(\log k)}\ n \log n$ time.*

▶ **Definition 35** (Coloring). *For a graph G, a* coloring *of G using k colors is a function $col : V(G) \to [k]$. A connected subgraph H of G is* colorful *if col is injective on $V(H)$.*

In the following lemma, we solve a sub-problem that we will use for our algorithm for PROP-ICFD using dynamic programming and color-coding.

▶ **Lemma 36** (∗). *Let G be a vertex-weighted graph with weight function $w : V(G) \to \mathbb{N}_0$. Then, given numbers $t, k \in \mathbb{N}$, we can decide in FPT time whether there exists a connected subgraph H of G such that $|V(H)| = k$ and $\sum_{v \in V(H)} w(v) \geq t$.*

Let $((G, A, \mathcal{U}, p), p)$ be an instance of PROP-ICFD. Let the vertices of G be colored using $|A|$ colors ($n = |A|$). A coloring is *suitable* if there exists a PROP allocation $\Pi = (\pi_1, \ldots, \pi_n)$ such that each vertex in π_i is colored with the color i. Now, we have the following lemma.

▶ **Lemma 37** (∗). *Given a coloring $col : V(G) \to A$ for an instance $(G, A, \mathcal{U}, p)$ of* PROP-ICFD*, we can decide whether this is a suitable coloring for $(G, A, \mathcal{U}, p)$ in $e^p p^{\mathcal{O}(p \log p)} m^{\mathcal{O}(1)}$ time.*

Next, we establish the probability of getting a suitable coloring of G, using $|A|$ colors uniformly at random, given that $(G, A, \mathcal{U}, p)$ is a Yes-instance of PROP-ICFD, in the following lemma.

▶ **Lemma 38** (∗). *If $(G, A, \mathcal{U}, p)$ is a Yes-instance of* PROP-ICFD*, then coloring the vertices of G uniformly at random using $|A|$ colors returns a suitable coloring with probability at least $(\frac{1}{|A|})^p$.*

Finally, we have the following theorem.

▶ **Theorem 5.** *There exists a randomized algorithm that solves* PROP-ICFD*, in time* $e^p p^{\mathcal{O}(p \log p)} m^{\mathcal{O}(1)}$ *with success probability at least* $1 - \frac{1}{e}$.

Proof Sketch. Using Lemma 37 and 38, we have an algorithm that given a Yes-instance, returns a solution with probability at least $\frac{1}{n^p}$. Clearly, by repeating this algorithm n^p times, the success probability of our algorithm is least $1 - \frac{1}{e}$. Since one execution of Lemma 37 requires $e^p p^{\mathcal{O}(p \log p)} m^{\mathcal{O}(1)}$ time and we repeat it $n^p \leq p^p$ times (since $n \leq p$), we get the desired running time. ◀

7 Conclusion

In this paper, we defined ICFD, a generalization of CFD, and conducted a comprehensive parameterized analysis of the problem. On the one hand, we proved that for $\varphi \in \{\mathsf{EF}, \mathsf{EF1}, \mathsf{EFX}\}$, φ-ICFD remains W[1]-hard parameterized by $p + \mathsf{vcn} + |A|$, even when $\mathsf{vcn} + |A|$ is a small constant. On the other hand, we proved that PROP-ICFD admits a randomized FPT algorithm parameterized by p alone. We also presented positive as well as negative results on the kernelization complexity of ICFD (for all fairness notions considered) parameterized by $\mathsf{vcn} + \mathsf{val} + p$.

There are several other generalizations of ICFD (or CFD) that can be practically well-motivated as well as theoretically interesting. For instance, the requirement of connectivity can be "relaxed" (for example, by requiring that each agent receives at most c connected components where $c \geq 1$) as well as "restricted" (for example, by requiring that each agent should receive a connected component that is 2-connected, or, more generally, c-connected where $c \geq 1$). Moreover, it might be desirable sometimes that the diameter of each connected component (or the sum of diameters of all connected components) assigned to an agent is bounded by a parameter d. For the examples we have already described in the Introduction, it is easy to see that such variants make sense.

Furthermore, several other notions of *fairness*, *welfare*, and *chores* were studied with respect to classical Fair Division. Hence, it might be interesting to consider ICFD in light of these notions. Moreover, it might be interesting to study ICFD by considering the parameter $|V(G) - p|$ along with other parameters. This might be specifically applicable when the allocator wants to allocate as much resources as possible.

In the classical fair division (ICFD on a clique with $p = |V(G)|$), it is known that EF1 always exists and can be computed in polynomial time using the famous envy-cycles procedure [30]. On the other hand, the guarantee of the existence of EFX exists for very restricted settings (for example, 4 agents with additive valuations [5] and 2 agents with arbitrary valuations [32]). It will be interesting to study if this difference in the "difficulty" of the two notions extends to the setting of ICFD and/or to the setting of CFD.

Finally, we discuss a natural variant of PROP-ICFD, namely PRP-ICFD, where every agent measures its happiness not in the context of its valuation of the whole graph G, but the subgraph G' of G that is assigned to the agents. More formally, an allocation $\Pi = (\pi_1, \ldots, \pi_n)$ respects PRP-ICFD if for every $i \in A$, $u_i(\pi_i) \geq \frac{u_i(V(G'))}{n}$, where $G' = G[\pi_1 \cup \cdots \cup \pi_n]$. Observe that, for PRP-ICFD, RR 1 is safe, and is sufficient to get an exponential kernel parameterized $\mathsf{vcn} + \mathsf{val} + p$, unlike PROP-ICFD, which requires more careful reduction rules. Moreover, the color-coding based FPT algorithm for PROP-ICFD is unlikely to work for PRP-ICFD since the key ingredient of this algorithm is that we know beforehand the exact valuation that will make an agent "happy", which is not the case for PRP-ICFD. In this regard, PRP-ICFD behaves more like EF, EF1, and EFX ICFD. Hence, it will be interesting to figure out whether PRP-ICFD is FPT or W[1]-hard when parameterized by $p + \mathsf{vcn}$.

References

1. Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *European Symposium on Algorithms*, pages 1–12. Springer, 2014.
2. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995. doi:10.1145/210332.210337.
3. Haris Aziz, Sylvain Bouveret, Ioannis Caragiannis, Ira Giagkousi, and Jérôme Lang. Knowledge, fairness, and social constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
4. Siddharth Barman and Sanath Kumar Krishnamurthy. Approximation algorithms for maximin fair division. *ACM Transactions on Economics and Computation (TEAC)*, 8(1):1–28, 2020.
5. Ben Berger, Avi Cohen, Michal Feldman, and Amos Fiat. Almost full efx exists for four agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4826–4833, 2022.
6. 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. *Games and Economic Behavior*, 131:197–221, 2022.
7. Bernhard Bliem, Robert Bredereck, and Rolf Niedermeier. Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In *IJCAI*, pages 102–108, 2016.
8. Niclas Boehmer, Robert Bredereck, Klaus Heeger, Dusan Knop, and Junjie Luo. Multivariate algorithmics for eliminating envy by donating goods. In Piotr Faliszewski, Viviana Mascardi, Catherine Pelachaud, and Matthew E. Taylor, editors, *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pages 127–135. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022. doi:10.5555/3535850.3535866.
9. Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. Fair division of a graph. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 135–141. AAAI Press, 2017.
10. Sylvain Bouveret and Jérôme Lang. Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. *J. Artif. Int. Res.*, 32(1):525–564, June 2008.
11. Sylvain Bouveret and Michel Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems*, 30(2):259–290, 2016.
12. Steven J Brams, Paul H Edelman, and Peter C Fishburn. Fair division of indivisible items. *Theory and Decision*, 55(2):147–180, 2003.
13. Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.
14. Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.
15. Ioannis Caragiannis, Nick Gravin, and Xin Huang. Envy-freeness up to any item with high nash welfare: The virtue of donating items. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 527–545, 2019.
16. Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32, 2019.
17. Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. Efx exists for three agents. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 1–19, 2020.
18. Bhaskar Ray Chaudhury, Telikepalli Kavitha, Kurt Mehlhorn, and Alkmini Sgouritsa. A little charity guarantees almost envy-freeness. *SIAM Journal on Computing*, 50(4):1336–1358, 2021.
19. Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

20 Argyrios Deligkas, Eduard Eiben, Robert Ganian, Thekla Hamm, and Sebastian Ordyniak. The parameterized complexity of connected fair division. In *IJCAI*, pages 139–145, 2021.

21 Argyrios Deligkas, Eduard Eiben, Robert Ganian, Thekla Hamm, and Sebastian Ordyniak. The complexity of envy-free graph cutting. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 237–243, 2022.

22 Reinhard Diestel. *Graph theory.* Springer Publishing Company, Incorporated, 2018.

23 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and ids. In *International Colloquium on Automata, Languages, and Programming*, pages 378–389. Springer, 2009.

24 Eduard Eiben, Robert Ganian, Thekla Hamm, and Sebastian Ordyniak. Parameterized complexity of envy-free resource allocation in social networks. *Artificial Intelligence*, 315:103826, 2023.

25 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing.* Cambridge University Press, 2019.

26 Harmender Gahlawat, , and Meirav Zehavi. Kernels for the disjoint paths problem on subclasses of chordal graphs. *arXiv*, 2023. `arXiv:2310.01310`.

27 Gianluigi Greco and Francesco Scarcello. The complexity of computing maximin share allocations on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2006–2013, 2020.

28 Ayumi Igarashi and Dominik Peters. Pareto-optimal allocation of indivisible goods with connectivity constraints. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 2045–2052, 2019.

29 David Kurokawa, Ariel D Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)*, 65(2):1–27, 2018.

30 Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131, 2004.

31 Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191. IEEE, 1995.

32 Benjamin Plaut and Tim Roughgarden. Almost envy-freeness with general valuations. *SIAM Journal on Discrete Mathematics*, 34(2):1039–1068, 2020.

33 Tayfun Sönmez and M Utku Ünver. Course bidding at business schools. *International Economic Review*, 51(1):99–123, 2010.

34 David P Williamson and David B Shmoys. *The design of approximation algorithms.* Cambridge university press, 2011.

Regular Separators for VASS Coverability Languages

Chris Köcher
Max Planck Institute for Software Systems, Kaiserslautern, Germany

Georg Zetzsche
Max Planck Institute for Software Systems, Kaiserslautern, Germany

Abstract

We study regular separators of vector addition systems (VASS, for short) with coverability semantics. A regular language R is a *regular separator* of languages K and L if $K \subseteq R$ and $L \cap R = \emptyset$. It was shown by Czerwiński, Lasota, Meyer, Muskalla, Kumar, and Saivasan (CONCUR 2018) that it is decidable whether, for two given VASS, there exists a regular separator. In fact, they show that a regular separator exists if and only if the two VASS languages are disjoint. However, they provide a triply exponential upper bound and a doubly exponential lower bound for the size of such separators and leave open which bound is tight.

We show that if two VASS have disjoint languages, then there exists a regular separator with at most doubly exponential size. Moreover, we provide tight size bounds for separators in the case of fixed dimensions and unary/binary encodings of updates and NFA/DFA separators. In particular, we settle the aforementioned question.

The key ingredient in the upper bound is a structural analysis of separating automata based on the concept of *basic separators*, which was recently introduced by Czerwiński and the second author. This allows us to determinize (and thus complement) without the powerset construction and avoid one exponential blowup.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Regular languages

Keywords and phrases Vector Addition System, Separability, Regular Language

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.15

Funding Funded by the European Union (ERC, FINABIS, 101077902). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

1 Introduction

Safety verification of concurrent systems typically consists of deciding whether two languages $K, L \subseteq \Sigma^*$ are disjoint: If each of the languages describes the set of event sequences that (i) are consistent with the behavior of a system component and (ii) reach an undesirable state, then their intersection is exactly the set of event sequences that are consistent with both components and reach the undesirable state.

If we wish to not only decide, but *certify* disjointness of languages $K, L \subseteq \Sigma^*$, then a natural kind of certificate is a *regular separator*: a regular language $R \subseteq \Sigma^*$ such that $K \subseteq R$ and $L \cap R = \emptyset$. Regular separators can indeed act as disjointness certificates: Deciding whether a given language intersects (resp. is included in) a regular language is usually simple.

The *regular separability* problem asks whether for two given languages there exists a regular separator. This decision problem has recently attracted a significant amount of interest. After the problem was shown to be undecidable for context-free languages in the 1970s [18, 33], recent work has a strong focus on *vector addition systems* (VASS), which

© Chris Köcher and Georg Zetzsche;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 15; pp. 15:1–15:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

are automata with counters that can be incremented, decremented, but not tested for zero. Typically, VASS are considered with two possible semantics: With the *reachability semantics*, where a target configuration has to be reached exactly, and the *coverability semantics*, where the target only has to be covered. Decidability of regular separability remains an open problem for reachability semantics. However, decidability has been established for coverability languages of VASS [10] and several other subclasses, such as one-dimensional VASS [9], integer VASS [6] (where counters can become negative), and commutative VASS languages [7]. Moreover, for each of these subclasses, decidability is retained if one of the input languages is an arbitrary VASS reachability language [13].

The decidability result about VASS coverability languages is a consequence of a remarkable and surprising result by Czerwiński, Lasota, Meyer, Muskalla, Kumar, and Saivasan [10]: Two languages of finitely-branching well-structured transition systems (WSTS) are separable by a regular language if and only if they are disjoint. (In fact, very recently, Keskin and Meyer [20] have even shown that the finite branching assumption is not required.) Moreover, VASS (with coverability semantics) are a standard example of (finitely branching) WSTS.

Despite this range of work on decidability, very little is known about a fundamental aspect of the separators: *What is the size of the separator, if they exist?* Here, by size, we mean the number of states in an NFA or DFA. In fact, the only result we are aware of is a partial answer for VASS coverability languages: In [10] a triply exponential upper bound and a doubly exponential lower bound is shown for NFA separating VASS coverability languages, leaving open whether there always exists a doubly-exponential separator.

Contribution. We study the size of regular separators in VASS coverability languages. Our first main result is that if two VASS coverability languages are disjoint, then there exists a doubly exponential-sized separating NFA. We then provide a comprehensive account of separator sizes for VASS languages: We study separator sizes in (i) fixed/arbitrary dimension, (ii) with unary/binary counter updates and (iii) deterministic/non-deterministic separators. In each case, we provide a tight polynomial or singly, doubly, or triply exponential bound.

Related work. There also exists some work on separability of relations by recognizable relations [1, 5] (which, in some precise sense, is also an instance of regular separability).

The equivalence between regular separability and disjointness for WSTS [10, 20] and the fact that decidability of the two problems usually coincide, raise the question of whether they are inter-reducible in general. However, there are language classes where disjointness is decidable and regular separability is undecidable [21, 34] and vice-versa [34].

Decidability of separability by piecewise testable languages is quite well understood. There is a language theoretic characterization [12] (which also holds for more general separator classes [35]) and a more abstract characterization (that also applies to trees) [15] of when separability is decidable.

There is long line of work on separability of regular languages of finite words by languages from smaller subclasses [11, 23–31]. Beyond finite words, separability has been studied for languages of infinite words (for regular languages [17] and Büchi VASS [2]) and for regular languages of finite trees [3] and infinite trees [8].

2 Preliminaries

Let $d \in \mathbb{N}^+$ be a positive number. A *vector* $\vec{v}$ over $\mathbb{Z}$ is an element $\vec{v} \in \mathbb{Z}^d$. For a vector $\vec{v} = (v_1, \ldots, v_d) \in \mathbb{Z}^d$ and a number $1 \leq i \leq d$ we write $\vec{v}[i]$ for the i-th component v_i of $\vec{v}$. By $\vec{0} \in \mathbb{Z}^d$ we denote the *zero vector* satisfying $\vec{0}[i] = 0$ for each $1 \leq i \leq d$. For two vectors

$\vec{u}, \vec{v} \in \mathbb{Z}^d$ we write $\vec{u} + \vec{v}$ for the vector $\vec{w} \in \mathbb{Z}^d$ with $\vec{w}[i] = \vec{u}[i] + \vec{v}[i]$ for each $1 \leq i \leq d$, i.e., $+$ is the component-wise addition. We write $\vec{u} \leq \vec{v}$ if, and only if, we have $\vec{u}[i] \leq \vec{v}[i]$ (for the natural ordering in $\mathbb{Z}$) for each $1 \leq i \leq d$. Note that $\leq$ is a partial ordering on $\mathbb{Z}^d$, but in the case of $d > 1$ no linear ordering.

Now, let $c, d \in \mathbb{N}^+$, $\vec{u} \in \mathbb{Z}^c$, and $\vec{v} \in \mathbb{Z}^d$. By $(\vec{u}, \vec{v})$ we denote the vector $\vec{w} \in \mathbb{Z}^{c+d}$ with $\vec{w}[i] = \vec{u}[i]$ for each $1 \leq i \leq c$ and $\vec{w}[i+c] = \vec{v}[i]$ for each $1 \leq i \leq d$, i.e., $(\vec{u}, \vec{v})$ is the concatenation of $\vec{u}$ and $\vec{v}$.

Vector Addition Systems. Let $d \in \mathbb{N}^+$. A *(d-dimensional) vector addition system with states* or *(d-)VASS* is a tuple $\mathfrak{V} = (Q, \Sigma, \Delta, s, t)$ where Q is a finite set of *states*, Σ is an alphabet, $\Delta \subseteq Q \times \Sigma_\varepsilon \times \mathbb{Z}^d \times Q$ is a finite set of transitions, and $s, t \in Q$ are its *source* resp. *target states*. Here, Σ_ε denotes the set $\Sigma \cup \{\varepsilon\}$. The vector $\vec{x} \in \mathbb{Z}^d$ of a transition $(p, a, \vec{x}, q) \in \Delta$ is called the *counter update* of this transition.

A *pseudo-configuration* is a tuple from $Q \times \mathbb{Z}^d$; it is called a *configuration* if this tuple is even contained in $Q \times \mathbb{N}^d$. A *pseudo-run* is a sequence $(q_i, \vec{v_i})_{0 \leq i \leq \ell}$ of pseudo-configurations such that for each $1 \leq i \leq \ell$ there is a transition $(q_{i-1}, a_i, \vec{x_i}, q_i) \in \Delta$ with $\vec{v_i} = \vec{v_{i-1}} + \vec{x_i}$. The *label* of such pseudo-run is $a_1 a_2 \ldots a_\ell \in \Sigma^*$; its *length* is ℓ (note that due to ε-labeled transitions we have $\ell \geq |a_1 a_2 \ldots a_\ell|$). A pseudo-run is called a *run* if we have $\vec{v_i} \in \mathbb{N}^d$ for each $0 \leq i \leq n$, i.e., if each intermediate pseudo-configuration is actually a configuration. For two configurations $(p, \vec{u}), (q, \vec{v}) \in Q \times \mathbb{N}^d$ and $w \in \Sigma^*$ we write $(p, \vec{u}) \xrightarrow{w}_{\mathfrak{V}} (q, \vec{v})$ if there is a run $(q_i, \vec{v_i})_{0 \leq i \leq \ell}$ with label w, $(p, \vec{u}) = (q_0, \vec{v_0})$, and $(q, \vec{v}) = (q_\ell, \vec{v_\ell})$. For a natural number $\ell \in \mathbb{N}$ we write $(p, \vec{u}) \rightarrow^\ell_{\mathfrak{V}} (q, \vec{v})$ if there is a run from $(p, \vec{u})$ to $(q, \vec{v})$ of length ℓ. We write $(p, \vec{u}) \rightarrow_{\mathfrak{V}} (q, \vec{v})$ if there exists such an ℓ.

The *(coverability) language* of $\mathfrak{V}$ is $\mathrm{L}(\mathfrak{V}) = \{w \in \Sigma^* \mid \exists \vec{v} \in \mathbb{N}^d \colon (s, \vec{0}) \xrightarrow{w}_{\mathfrak{V}} (t, \vec{v})\}$ (note that $\vec{v} \geq \vec{0}$ holds for any $\vec{v} \in \mathbb{N}^d$; we say that $(t, \vec{v})$ *covers* the so-called target configuration $(t, \vec{0})$). We say $L \subseteq \Sigma^*$ is a *(coverability) d-VASS-language* if there is a d-VASS $\mathfrak{V}$ with $L = \mathrm{L}(\mathfrak{V})$.

Now, let $\mathfrak{V}_i = (Q_i, \Sigma, \Delta_i, s_i, t_i)$ be two d-VASS ($i = 1, 2$). We want to construct the product VASS $\mathfrak{V}_1 \times \mathfrak{V}_2$ which simulates $\mathfrak{V}_1$ and $\mathfrak{V}_2$ in parallel. To this end, set the $2d$-VASS $\mathfrak{V}_1 \times \mathfrak{V}_2 := (Q_1 \times Q_2, \Sigma, \Delta, (s_1, s_2), (t_1, t_2))$ with the following transitions in Δ:

- $((p_1, p_2), a, (\vec{v_1}, \vec{v_2}), (q_1, q_2)) \in \Delta$ if $(p_1, a, \vec{v_1}, q_1) \in \Delta_1$ and $(p_2, a, \vec{v_2}, q_2) \in \Delta_2$,
- $((p_1, p_2), \varepsilon, (\vec{v_1}, \vec{0}), (q_1, p_2)) \in \Delta$ if $(p_1, \varepsilon, \vec{v_1}, q_1) \in \Delta_1$, and
- $((p_1, p_2), \varepsilon, (\vec{0}, \vec{v_2}), (p_1, q_2)) \in \Delta$ if $(p_2, \varepsilon, \vec{v_2}, q_2) \in \Delta_2$.

Then the following statement is easy to see:

▶ **Lemma 2.1.** *Let $\mathfrak{V}_1$ and $\mathfrak{V}_2$ be two d-VASS. Then $\mathrm{L}(\mathfrak{V}_1 \times \mathfrak{V}_2) = \mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2)$, i.e., the intersection of two d-VASS-languages is a 2d-VASS-language.*

For a vector $\vec{v} \in \mathbb{Z}^d$ let $\|\vec{v}\| = \max\{|\vec{v}[i]| \colon 1 \leq i \leq d\}$ be the *norm* of $\vec{v}$ (where $|x|$ is the absolute value of $x \in \mathbb{Z}$). We also define the norm of the transition relation Δ as follows: $\|\Delta\| := \max\{\|\vec{v}\| \colon (p, a, \vec{v}, q) \in \Delta\}$. Then the *size* $|\mathfrak{V}|$ of the d-VASS $\mathfrak{V}$ is $|Q| + d \cdot |\Delta| \cdot \|\Delta\|$.

We can define the *Rackoff-number* $\mathrm{Rack}(\mathfrak{V})$ of $\mathfrak{V}$: $\mathrm{Rack}(\mathfrak{V}) := (|Q| \cdot \|\Delta\| + 2)^{(3d)!+1}$. Then we can show that for each run from a configuration $c \in Q \times \mathbb{N}^d$ covering the target configuration $(t, \vec{0})$ there is also such run of length bounded by the Rackoff-number. This is the following central statement:

▶ **Theorem 2.2** ([4,32]). *Let $\mathfrak{V} = (Q, \Sigma, \Delta, s, t)$ be a d-VASS and $c \in Q \times \mathbb{N}^d$ be a configuration such that there is a vector $\vec{v} \in \mathbb{N}^d$ with $c \rightarrow_{\mathfrak{V}} (t, \vec{v})$. Then there are $\ell \in \mathbb{N}$ and $\vec{w} \in \mathbb{N}^d$ with $0 \leq \ell \leq \mathrm{Rack}(\mathfrak{V})$ and $c \rightarrow^\ell_{\mathfrak{V}} (t, \vec{w})$.*

Table 1 An overview of the (matching) upper and lower bounds for finite automata separating two disjoint d-VASS. We distinguish between (i) whether the dimension $d \in \mathbb{N}^+$ is part of the input, (ii) whether the separating automaton should be an NFA or a DFA, and (iii) whether counter updates are encoded in unary or binary. The colors denote the employed lower bound technique.

		NFAs		DFAs	
		unary	binary	unary	binary
d as input		2-exp.	2-exp.	3-exp.	3-exp.
d fixed	$d \geq 2$	poly.	exp.	exp.	2-exp.
	$d = 1$	poly.	exp.	exp.	exp.

The bound above is due to Bozelli and Ganty [4], which is slightly tighter than Rackoff's original bound of $2^{2^{\mathcal{O}(\|\Delta\| \log \|\Delta\|)}}$ [32]. It should be noted that very recently, a significantly better upper bound has been obtained [22]).

Regular languages. A *non-deterministic finite automaton* or *NFA* is a tuple $\mathfrak{A} = (Q, \Sigma, \delta, I, F)$ where Q is a finite set of *states*, Σ is an alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is a set of *transitions*, and $I, F \subseteq Q$ are the sets of *initial* resp. *accepting* states. It is called *deterministic* or *DFA* if $|I| = 1$ and for each $p \in Q$ and $a \in \Sigma$ there is exactly one $q \in Q$ with $(p, a, q) \in \delta$. The size $|\mathfrak{A}|$ of $\mathfrak{A}$ is $|Q|$. For $p, q \in Q$ and $w \in \Sigma^*$ we write $p \xrightarrow{w}_{\mathfrak{A}} q$ if there are $a_1, \ldots, a_\ell \in \Sigma$ and $q_0, q_1, \ldots, q_\ell \in Q$ with $w = a_1 a_2 \ldots a_\ell$, $p = q_0$, $q = q_\ell$, and $(q_{i-1}, a_i, q_i) \in \delta$ for each $1 \leq i \leq \ell$. The *accepted language* of $\mathfrak{A}$ is $\mathrm{L}(\mathfrak{A}) = \{w \in \Sigma^* \mid \exists \iota \in I, f \in F \colon \iota \xrightarrow{w}_{\mathfrak{A}} f\}$. A language $L \subseteq \Sigma^*$ is called *regular* if there is an NFA $\mathfrak{A}$ with $L = \mathrm{L}(\mathfrak{A})$.

Regular Separability. Let Σ be an alphabet. Two languages $K, L \subseteq \Sigma^*$ are called *regular separable* (denoted $K \mid L$) if there is a regular language $R \subseteq \Sigma^*$ with $K \subseteq R$ and $L \cap R = \emptyset$. In this case R is called a *regular separator* of K and L. We say that any NFA accepting R *separates* K and L. Since the class of regular languages is closed under complement, we learn that if $K \mid L$ holds, then also $L \mid K$ (via the complementary separator).

The following equivalence is known about the languages of coverability VASS. Note that actually Czerwiński et al. [10] have shown this result for the languages of a more general notion – so-called *well structured transition systems* (or *WSTS* for short, cf. e.g. [14]).

▶ **Theorem 2.3** ([10]). *Let $\mathfrak{V}$ and $\mathfrak{W}$ be two VASS. Then we have $\mathrm{L}(\mathfrak{V}) \mid \mathrm{L}(\mathfrak{W})$ if, and only if, $\mathrm{L}(\mathfrak{V}) \cap \mathrm{L}(\mathfrak{W}) = \emptyset$.*

3 Main Results

In this section, we present the main results of this work. An overview can be found in Table 1. Here, by i-exp, we mean that there is an i-fold exponential upper bound. More precisely, there exists a separator with at most $\exp_i(\mathrm{poly}(n))$ states for input VASS of size n. Here $\exp_0(n) = n$ and $\exp_{i+1}(n) = 2^{\exp_i(n)}$ for $i \geq 0$. All our bounds are tight in the sense that for each i-fold exponential upper bound with $i \geq 1$, we present a sequence of VASS pairs of size polynomial in n such that the smallest separator requires $\exp_i(n)$ states. Proofs can be found in Sections 5 and 6 (upper bounds and lower bounds, resp.).

First upper bound. Our first upper bound result is the following.

▶ **Theorem 3.1.** *Let $\mathfrak{V}_1$ and $\mathfrak{V}_2$ be d-VASS with at most $n \geq 1$ states and updates of norm at most $m \geq 1$. If $\mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2) = \emptyset$, then $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ are separated by an NFA with at most $(n+m)^{2^{\mathrm{poly}(d)}}$ states.*

This provides almost all upper bounds in Table 1. In particular, it closes the gap left by [10] by providing a doubly exponential upper bound for NFA separators in the general case.

Let us explain how we avoid one exponential blow-up compared to [10]. In [10], the authors first construct VASS $\mathfrak{V}_1'$ and $\mathfrak{V}_2'$ such that (i) $\mathfrak{V}_2'$ is deterministic, (ii) $\mathrm{L}(\mathfrak{V}_1') \cap \mathrm{L}(\mathfrak{V}_2') = \emptyset$ and (iii) any separator for $\mathrm{L}(\mathfrak{V}_1')$ and $\mathrm{L}(\mathfrak{V}_2')$ can be transformed into a separator for $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$. Then, relying on Rackoff-style bounds for covering runs in VASS, they construct a doubly exponential NFA separator for $\mathrm{L}(\mathfrak{V}_1')$ and $\mathrm{L}(\mathfrak{V}_2')$. The latter step yields an inherently non-deterministic separator. However, the transformation mentioned in (iii) requires a complementation, which results in a triply exponential bound overall.

Instead, roughly speaking, we first apply an observation from [13] to reduce to an even more specific case: We construct $\mathfrak{V}$ such that for the language C_d of all counter instruction sequences that keep the d counters above zero, we have (a) $\mathrm{L}(\mathfrak{V}) \cap C_d = \emptyset$ and (b) any separator of $\mathrm{L}(\mathfrak{V})$ and C_d can be transformed into a separator for $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$. Then, we rely on the fact that a particular family $(B_k)_{k \in \mathbb{N}}$ of regular languages is a family of *basic separators* (a concept introduced by Czerwiński and the second author in [13]): Every language regularly separable from C_d is included in a finite union of sets B_k. Here, B_k contains all sequences of counter instructions such that at least one counter at some point falls below zero, but before that, it never exceeds the value k. We prove a version of this with complexity bounds: We show that $\mathrm{L}(\mathfrak{V}) \cap C_d = \emptyset$ implies that $\mathrm{L}(\mathfrak{V})$ is included in B_k for some doubly exponential bound k. Here, the key advantage is that we understand the structure of the B_k so well that we can just observe that the separator B_k is already deterministic. Thus, the complementation step will not result in another exponential blow-up.

Second upper bound. Theorem 3.1 provides all upper bounds for NFA separators in Table 1. It also provides all upper bounds for DFAs where the DFA bound is exponential in the corresponding NFA bound (via the powerset construction). The only exception to this is the dark gray entry: Here, the tight DFA bound is actually the same as for NFA.

▶ **Theorem 3.2.** *Let $\mathfrak{V}_1$ and $\mathfrak{V}_2$ be 1-VASS with binary updates. If $\mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2) = \emptyset$, then there exists a separating DFA with at most exponentially many states.*

For this, we observe that the states of NFA resulting from Theorem 3.1 for $d = 1$ can be equipped with a partial ordering $\leq$ such that (i) if $p \leq q$, then all words accepted from p are also accepted from q and (ii) every anti-chain in this ordering has at most polynomial size. This permits determinization without a blow-up.

Lower bounds. The lower bounds for the first row of Table 1 are known from [10]. For the others, we use two types of pairs. The first is similar to the language pairs in [10]:

$$\begin{aligned} K_{f,n} &= \{w \in \{a,b\} \mid \text{the } f(n)\text{-th last letter of } w \text{ is an } a \text{ and } |w| \geq f(n)\} \\ L_{f,n} &= \{w \in \{a,b\} \mid \text{the } f(n)\text{-th last letter of } w \text{ is a } b \text{ or } |w| < f(n)\} \end{aligned} \tag{1}$$

where $f \colon \mathbb{N} \to \mathbb{N}$ is one of the functions $n \mapsto n$ (a separating DFA needs 2^n states; the blue entries) or $n \mapsto 2^n$ (a separating DFA needs 2^{2^n} states, the yellow entry). In [10], these are used for $n \mapsto 2^{2^n}$. The second language pair consists of $L_n = \{a^m \mid m \geq 2^n\}$, and $K_n = \{a^m \mid m < 2^n\}$ (an NFA needs 2^n states, the light and dark gray entries).

4 Basic Separators

As already mentioned in the previous section we want to apply the approach from [13] to show our main theorem. To this end, we first have to introduce languages following the courses of the counters of our VASS. We introduce two (basic) actions a_i and $\overline{a_i}$ for each $1 \leq i \leq d$ to indicate that counter i gets increased resp. decreased by one. By $\Gamma_d := \{a_i, \overline{a_i} \mid 1 \leq i \leq d\}$ we denote the alphabet of basic actions. Then a word $w \in \Gamma_d^*$ encodes the course of updates of the d counters on some pseudo-run of a d-VASS. For $1 \leq i \leq d$ we introduce a homomorphism $\phi_i \colon \Gamma_d^* \to \mathbb{Z}$ induced by the equations $\phi_i(a_i) = 1$, $\phi_i(\overline{a_i}) = -1$, and $\phi_i(b) = 0$ for $b \in \Gamma_d \setminus \{a_i, \overline{a_i}\}$. In other words, $\phi_i(w)$ is the value of counter i after application of the actions specified in w.

For $w \in \Gamma_d^*$ define $\mathrm{drop}_i(w) := \min\{\phi_i(v) \mid v \text{ is a prefix of } w\} \in [-|w|, 0]$, i.e., $\mathrm{drop}_i(w)$ is the lowest value the counter i had while applying the actions in w. In a run counter i starts with value 0 and stays non-negative. Therefore, any run $w \in \Gamma_d^*$ of a d-VASS satisfies $\mathrm{drop}_i(w) = 0$. By $G_i := \{w \in \Gamma_d^* \mid \mathrm{drop}_i(w) = 0\}$ we define the language of all action sequences where counter i never falls below zero. Then the language of all runs of a d-VASS is $C_d := \bigcap_{i=1}^{d} G_i$. Next, we want to describe the courses $w \in \Gamma_d^*$ of pseudo-runs of a given VASS $\mathfrak{V}$. To this end, we first have to recall the notion of rational transductions:

Rational Transductions. Let Σ and Γ be two alphabets. A *transducer* is a tuple $\mathfrak{T} = (Q, \delta, I, F)$ where Q is a finite set of *states*, $\delta \subseteq Q \times \Gamma^* \times \Sigma^* \times Q$ is a finite set of *transitions*, and $I, F \subseteq Q$ are the *initial* resp. *accepting* states. A pair $(v, w) \in \Gamma^* \times \Sigma^*$ is *accepted* by $\mathfrak{T}$ if there are $q_0, q_1, \ldots, q_n \in Q$, $v_1, \ldots, v_n \in \Sigma^*$, and $w_1, \ldots, w_n \in \Gamma^*$ with $v = v_1 \ldots v_n$, $w = w_1 \ldots w_n$, $q_0 \in I$, $q_n \in F$, and $(q_{i-1}, v_i, w_i, q_i) \in \delta$ for each $1 \leq i \leq n$. The *accepted relation* of $\mathfrak{T}$ is $\mathrm{R}(\mathfrak{T}) = \{(v, w) \in \Gamma^* \times \Sigma^* \mid (v, w) \text{ is accepted by } \mathfrak{T}\}$. A relation $T \subseteq \Gamma^* \times \Sigma^*$ is called a *rational transduction* if there is a transducer $\mathfrak{T}$ with $\mathrm{R}(\mathfrak{T}) = T$. For a relation $T \subseteq \Gamma^* \times \Sigma^*$ and a language $L \in \Gamma^*$ we write $T(L)$ for the language $\{w \in \Sigma^* \mid \exists v \in L \colon (v, w) \in T\}$. Additionally, we write T^{-1} for the relation $\{(w, v) \in \Sigma^* \times \Gamma^* \mid (v, w) \in T\}$. The following connection between d-VASS and transducers is well-known:

▶ **Lemma 4.1** (cf. [16, 19]). *A language $L \subseteq \Sigma^*$ is a coverability d-VASS-language if, and only if, there is a rational transduction $T \subseteq \Gamma_d^* \times \Sigma^*$ with $L = T(C_d)$.*

Proof idea. We only show the implication "$\Rightarrow$" (for the converse implication cf. e.g. [13]). So, let $\mathfrak{V} = (Q, \Sigma, \Delta, s, t)$ be a d-VASS with $\mathrm{L}(\mathfrak{V}) = L$. We construct the following transducer $\mathfrak{T}_{\mathfrak{V}} = (Q, \delta, \{s\}, \{t\})$: set $\delta = \{(p, \mathrm{code}(\vec{v}), a, q) \mid (p, a, \vec{v}, q) \in \Delta\}$, where $\mathrm{code}(\vec{v}) = a_1^{\vec{v}[1]} a_2^{\vec{v}[2]} \ldots a_d^{\vec{v}[d]}$ and $a_i^n := \overline{a_i}^{|n|}$ holds for $n < 0$. Then we can see $(\mathrm{R}(\mathfrak{T}_{\mathfrak{V}}))(C_d) = \mathrm{L}(\mathfrak{V})$. ◀

Regular separability is, in some sense, compatible with rational transductions:

▶ **Lemma 4.2** ([13]). *Let $K \subseteq \Sigma^*$, $L \subseteq \Gamma^*$ be two languages and $T \subseteq \Gamma^* \times \Sigma^*$ be a rational transduction. Then $K \mid T(L)$ if, and only if, $T^{-1}(K) \mid L$.*

We include the (very simple) proof of this lemma, as it hints at our proof of Theorem 5.1:

Proof. For the "only if", suppose $K \mid T(L)$ with a regular separator $R \subseteq \Sigma^*$. It is easy to check that then, $T^{-1}(R) \subseteq \Gamma^*$ is a regular separator of $T^{-1}(K)$ and L. Thus $T^{-1}(K) \mid L$ holds. Conversely, assume $T^{-1}(K) \mid L$ via the regular separator $R \subseteq \Gamma^*$. Then we also know $L \mid T^{-1}(K)$ via $\Gamma^* \setminus R$. The proof of the "only if" direction yields $T(L) \mid K$ via $T(\Gamma^* \setminus R) \subseteq \Sigma^*$. Finally, we obtain $K \mid T(L)$ via $\Sigma^* \setminus T(\Gamma^* \setminus R)$. ◀

Basic separators. From Theorem 2.3 and Lemma 4.2 we learn that two d-VASS-languages $L, K \subseteq \Sigma^*$ are regular separable if, and only if, $T^{-1}(K) \mid C_d$ holds, where T is a rational transduction with $L = T(C_d)$. Czerwiński and the second author of this work have introduced in [13] the notion of *basic separators* of any language from the language C_d. These are families of regular languages disjoint from C_d such that each regular language, which is disjoint from C_d, is included in a finite union of basic separators. For coverability d-VASS suitable basic separators are the languages $B_k \subseteq \Gamma_d^*$ which contain all action sequences having one counter $1 \leq i \leq d$ falling below zero, but before that, counter i never exceeds the value of k. To this end, we first define the value $\mu_i(w) := \max\{\phi_i(v) \mid v \text{ is a prefix of } w \text{ with } \mathrm{drop}_i(v) = 0\}$ of a word $w \in \Gamma_d^*$. This is the greatest value of counter i before it falls below zero for the first time (or it is the maximal value of counter i if it always stays non-negative). Then B_k (for $k \in \mathbb{N}$) is defined as follows:

$$B_k := \{w \in \Gamma_d^* \mid \exists 1 \leq i \leq d \colon w \notin G_i \text{ and } \mu_i(w) \leq k\}\,.$$

As shown in [13], the following equivalence holds for coverability d-VASS:

▶ **Corollary 4.3.** *Let $\mathfrak{V}$ and $\mathfrak{W}$ be two d-VASS and $T \subseteq \Gamma_d^* \times \Sigma^*$ be a rational transduction with $\mathrm{L}(\mathfrak{W}) = T(C_d)$. Then the following properties are equivalent:*

1. $\mathrm{L}(\mathfrak{V}) \cap \mathrm{L}(\mathfrak{W}) = \emptyset$
2. $\mathrm{L}(\mathfrak{V}) \mid \mathrm{L}(\mathfrak{W})$
3. $T^{-1}(\mathrm{L}(\mathfrak{V})) \mid C_d$
4. *there is $k \in \mathbb{N}$ such that B_k is a regular separator of $T^{-1}(\mathrm{L}(\mathfrak{V}))$ and C_d.*

In the proof of our main result Theorem 5.1, we will show that there is a "small" $k \in \mathbb{N}$ such that B_k separates $T^{-1}(\mathrm{L}(\mathfrak{V}))$ and C_d.

5 Upper Bounds

We now prove Theorems 3.1 and 3.2. For Theorem 3.1, we prove a more concrete bound:

▶ **Theorem 5.1.** *Let $\mathfrak{V}_i = (Q_i, \Sigma, \Delta_i, s_i, t_i)$ (for $i = 1, 2$) be two d-VASS with $\mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2) = \emptyset$. Then there is an NFA of size at most $\mathcal{O}(|Q_1| \cdot \|\Delta_1\|^d \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)^d)$ separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$.*

This clearly implies Theorem 3.1. To show this, let $\mathfrak{V}_i = (Q_i, \Sigma, \Delta_i, s_i, t_i)$ (for $i = 1, 2$) be two disjoint d-VASS and let $\mathfrak{V}_1 \times \mathfrak{V}_2 = (Q_1 \times Q_2, \Sigma, \Delta, (s_1, s_2), (t_1, t_2))$ be the product VASS as constructed in Lemma 2.1. Note that $\mathrm{L}(\mathfrak{V}_1 \times \mathfrak{V}_2) = \emptyset$ holds due to the disjointness of $\mathfrak{V}_1$ and $\mathfrak{V}_2$. Additionally, let $\mathfrak{T}_{\mathfrak{V}_1}$ be the transducer constructed in the proof of Lemma 4.1 satisfying the property $(\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1}))(C_d) = \mathrm{L}(\mathfrak{V}_1)$. According to Corollary 4.3 the assumption $\mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2) = \emptyset$ implies $(\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1}))^{-1}(\mathrm{L}(\mathfrak{V}_2)) \mid C_d$. Our aim is to find a "small" number $\hat{k} \in \mathbb{N}$ such that $B_{\hat{k}}$ is a regular separator of $K(\mathfrak{V}_1, \mathfrak{V}_2) := (\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1}))^{-1}(\mathrm{L}(\mathfrak{V}_2))$ and C_d.

Let $w \in K(\mathfrak{V}_1, \mathfrak{V}_2) \subseteq \Gamma_d^*$ be some word. Then there is another word $w' \in \Sigma^*$ with $(w, w') \in R(\mathfrak{T}_{\mathfrak{V}_1})$ and $w' \in L(\mathfrak{V}_2)$. From this fact we obtain a w'-labeled pseudo-run of $\mathfrak{V}_1$ from s_1 to t_1 such that w encodes the counter updates of this pseudo-run. Additionally, we obtain a w'-labeled run of $\mathfrak{V}_2$ from s_2 to t_2. We can compose these two pseudo-runs to one w'-labeled pseudo-run of $\mathfrak{V}_1 \times \mathfrak{V}_2$ from (s_1, s_2) to (t_1, t_2). So, there is a sequence $\pi_w := ((p_i, q_i), (\vec{x_i}, \vec{y_i}))_{0 \leq i \leq n}$ of pseudo-configurations and transitions $((p_{i-1}, q_{i-1}), b_i, (\vec{u_i}, \vec{v_i}), (p_i, q_i)) \in \Delta$ (for $1 \leq i \leq n$) with $(p_0, q_0) = (s_1, s_2)$, $(p_n, q_n) = (t_1, t_2)$, $(\vec{x_0}, \vec{y_0}) = (\vec{0}, \vec{0})$, $(\vec{x_i}, \vec{y_i}) = (\vec{x_{i-1}} + \vec{u_i}, \vec{y_{i-1}} + \vec{v_i})$ for each $1 \leq i \leq n$, $w = \mathrm{code}(\vec{u_1}) \ldots \mathrm{code}(\vec{u_n})$ (note that this equation holds by the choice of our transducer $\mathfrak{T}_{\mathfrak{V}_1}$ from Lemma 4.1), and

$w' = b_1 b_2 \dots b_n$. Since we have $\mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2) = \emptyset$ by assumption, π_w is a pseudo-run of $\mathfrak{V}_1 \times \mathfrak{V}_2$, but actually not a run, i.e., at least one counter of $\mathfrak{V}_1 \times \mathfrak{V}_2$ falls below zero at some time. As stated above, w' labels a run of $\mathfrak{V}_2$, i.e., no counter of $\mathfrak{V}_2$ ever falls below zero. This implies the existence of $0 \leq i \leq n$ with $\vec{x_i} \in \mathbb{Z}^d \setminus \mathbb{N}^d$.

Set $\hat{k} := \|\Delta_1\| \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)$. By definition we know that $B_{\hat{k}} \cap C_d = \emptyset$ holds. So, we only have to prove $K(\mathfrak{V}_1, \mathfrak{V}_2) \subseteq B_{\hat{k}}$, i.e., we show that for each $w \in K(\mathfrak{V}_1, \mathfrak{V}_2)$ there is a counter $1 \leq i \leq d$ having a value at most $\hat{k}$ before falling below zero. We show this result by contradiction: assume there is $w \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$. This means, in w for each counter $1 \leq i \leq d$ we have two possibilities: (i) the counter i stays non-negative (i.e., $w \in G_i$) or (ii) the counter i falls below zero and before this happens for the first time it exceeds the value $\hat{k}$. We construct then another word $v \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$ having more counters $1 \leq i \leq d$ satisfying $v \in G_i$ than w. By induction we obtain a word $v \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$ with $v \in G_i$ for each $1 \leq i \leq d$. This implies $v \in C_d$ and therefore the existence of another word $v' \in \Sigma^*$ with $(v, v') \in R(\mathfrak{T}_{\mathfrak{V}_1})$ and $v' \in L(\mathfrak{V}_2)$. Then v' is the label of runs in $\mathfrak{V}_1$ (since $v \in C_d$) and $\mathfrak{V}_2$ (since $v' \in L(\mathfrak{V}_2)$). Hence, we obtain $v' \in \mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2)$, which is a contradiction to our assumption that $\mathfrak{V}_1$ and $\mathfrak{V}_2$ accept disjoint languages.

▶ **Lemma 5.2.** *We have $K(\mathfrak{V}_1, \mathfrak{V}_2) \subseteq B_{\hat{k}}$.*

To show this lemma we first have to introduce another notion: let $I \subseteq \{1, \dots, d\}$. For a vector $\vec{v} \in \mathbb{Z}^d$ we define the *projection* $\vec{v}^I$ to the components specified in I as follows: $\vec{v}^I[j] = \vec{v}[j]$ if $j \in I$ and $\vec{v}^I[j] = 0$ if $j \notin I$. Now, let $\mathfrak{V} = (Q, \Sigma, \Delta, s, t)$ be a d-VASS. For a pseudo-configuration $c = (q, \vec{v}) \in Q \times \mathbb{Z}^d$ we define $c^I := (q, \vec{v}^I)$. The *projection* $\mathfrak{V}^I$ of $\mathfrak{V}$ to I is the d-VASS $\mathfrak{V}^I = (Q, \Sigma, \Delta^I, s, t)$ with $\Delta^I := \{(p, a, \vec{v}^I, q) \mid (p, a, \vec{v}, q) \in \Delta\}$.

Proof. Let $w \in K(\mathfrak{V}_1, \mathfrak{V}_2)$. Towards a contradiction we suppose that $w \notin B_{\hat{k}}$ holds. Then for each $1 \leq i \leq d$ we have either $w \in G_i$ (i.e., the i-th counter never falls below 0), or $w \notin G_i$ and $\mu_i(w) > \hat{k}$ (i.e., the i-th counter reaches a value $> \hat{k}$ before falling below 0 for its first time). Let $I_w \subseteq \{1, \dots, d\}$ be the set of indices $1 \leq i \leq d$ with $w \in G_i$. Assuming $|I_w| < d$ we want to construct from w another word $v \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$ with $|I_v| > |I_w|$.

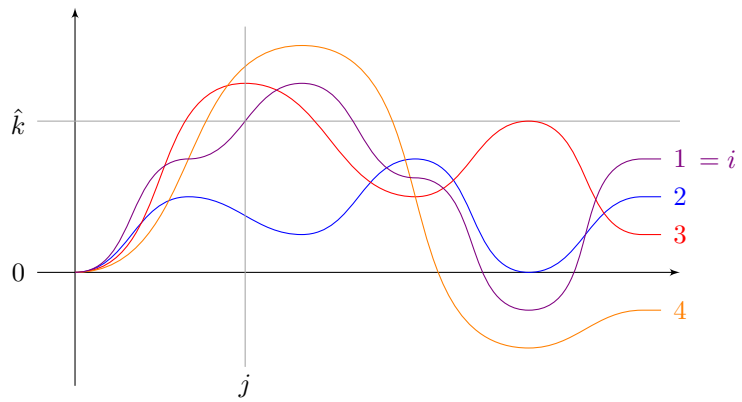


Figure 1 The values of the counters in the course of a run encoded by some word $w \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$. The values of the counters 2 and 3 never fall below zero implying $w \in G_2 \cap G_3$. The counters 1 and 4 exceed the value $\hat{k}$ before they fall below zero the first time (i.e., $w \notin G_1 \cup G_4$ and $\mu_1(w), \mu_4(w) > \hat{k}$). Since counter 1 exceeds $\hat{k}$ after counter 4 does, we choose $i = 1$ in our proof. The first intersection of counter 1's curve and $\hat{k}$ marks the step j.

So, let $i \in \{1, \dots, d\} \setminus I_w$ be the index of the last counter exceeding the upper bound $\hat{k}$ before it falls below zero for its first time, i.e., i is the number of the counter having the longest prefix w_1 of w with $\mu_i(w_1) \leq \hat{k}$. Additionally, let $0 \leq j < n$ be the first computational step in

which counter i exceeds $\hat{k}$, i.e., $\vec{x_j}[i] > \hat{k}$ and $\vec{x_h}[i] \leq \hat{k}$ for each $0 \leq h < j$. Now, restrict the pseudo-run π_w to the counters in I_w and all of $\mathfrak{V}_2$'s counters. Since none of these counters falls below zero, the pseudo-run π_w is actually a run in $\mathfrak{V}_1^{I_w} \times \mathfrak{V}_2$. This especially holds for π_w's sub-run from $((p_j, q_j), (\vec{x_j}^{I_w}, \vec{y_j}))$ to $((p_n, q_n), (\vec{x_n}^{I_w}, \vec{y_n}))$ in $\mathfrak{V}_1^{I_w} \times \mathfrak{V}_2$. Since $(p_n, q_n) = (t_1, t_2)$ holds, there is – according to Theorem 2.2 – also a run from $((p_j, q_j), (\vec{x_j}^{I_w}, \vec{y_j}))$ to some configuration $((p_n, q_n), (\vec{x'_m}^{I_w}, \vec{y'_m}))$ of length at most $\mathrm{Rack}(\mathfrak{V}_1^{I_w} \times \mathfrak{V}_2) \leq \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)$. Since $\mathfrak{V}_1^{I_w}$ is a projection of $\mathfrak{V}_1$ to the counters in I_w, we can also extend this run to a pseudo-run with all counters. Let $((p'_h, q'_h), (\vec{x'_h}, \vec{y'_h}))_{j \leq h \leq m}$ be this pseudo-run extended to all $2d$ counters satisfying $((p_j, q_j), (\vec{x_j}, \vec{y_j})) = ((p'_j, q'_j), (\vec{x'_j}, \vec{y'_j}))$, $p'_m = p_n = t_1$, and $q'_m = q_n = t_2$. Let $((p'_{h-1}, q'_{h-1}), b'_h, (\vec{u'_h}, \vec{v'_h}), (p'_h, q'_h)) \in \Delta$ be the corresponding transitions (for $j < h \leq m$). Set $v := \mathrm{code}(\vec{u_1}) \dots \mathrm{code}(\vec{u_j})\, \mathrm{code}(\vec{u'_{j+1}}) \dots \mathrm{code}(\vec{u'_m})$. Our next aim is to prove that $|I_v| > |I_w|$ and $v \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$ holds.

▷ Claim 5.3. $|I_v| > |I_w|$

Proof. We show $I_w \uplus \{i\} \subseteq I_v$. By the choice of our pseudo-run we have $I_w \subseteq I_v$ (recall that all counters from I_w always stay ≥ 0). So, we only have to show $i \in I_v$. We have $\vec{x_j}[i] = \vec{x'_j}[i] > \hat{k} = \|\Delta_1\| \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)$, $m - j \leq \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)$, and $\vec{u'_h}[i] \leq \|\Delta_1\|$ for each $j < h \leq m$. Hence, we obtain $\vec{x'_h}[i] \geq 0$ for each $j \leq h \leq m$, i.e., on our new run the counter i never falls below zero. We infer $v \in G_i$ and, therefore, $i \in I_v$. ◁

▷ Claim 5.4. $v \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$

Proof. First, we show $v \in K(\mathfrak{V}_1, \mathfrak{V}_2)$. The sequence

$$((p_{h-1}, q_{h-1}), b_h, (\vec{u_h}, \vec{v_h}), (p_h, q_h))_{1 \leq h \leq j}, ((p'_{h-1}, q'_{h-1}), b'_h, (\vec{u'_h}, \vec{v'_h}), (p'_h, q'_h))_{j < h \leq m}$$

of transitions in $\mathfrak{V}_1 \times \mathfrak{V}_2$ induces some accepting run $(q_0, \vec{0}) \xrightarrow{b_1 \dots b_j b'_{j+1} \dots b'_m}_{\mathfrak{V}_2} (q'_m, \vec{v'_m})$ in $\mathfrak{V}_2$, i.e., we have $b_1 \dots b_j b'_{j+1} \dots b'_m \in \mathrm{L}(\mathfrak{V}_2)$. Additionally, the word v encodes the counters' course of updates in the transition sequence $(p_{h-1}, b_h, \vec{u_h}, p_h)_{1 \leq h \leq j}, (p'_{h-1}, b'_h, \vec{u'_h}, p'_h)_{j < h \leq m}$ in $\mathfrak{V}_1$. According to $p_0 = s_1$, $p_j = p'_j$, and $p'_m = t_1$ this transition sequence is a pseudo-run of $\mathfrak{V}_1$ labeled by $b_1 \dots b_j b'_{j+1} \dots b'_m$. By the choice of our transducer $\mathfrak{T}_{\mathfrak{V}_1}$ (which is the one from the proof of Lemma 4.1), we learn $(v, b_1 \dots b_j b'_{j+1} \dots b'_m) \in \mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1})$ implying $v \in (\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1}))^{-1}(b_1 \dots b_j b'_{j+1} \dots b'_m)$. We finally obtain $v \in K(\mathfrak{V}_1, \mathfrak{V}_2)$.

Now, we prove $v \notin B_{\hat{k}}$. If we have $I_v = \{1, \dots, d\}$ then we learn $v \in \bigcap_{h=1}^{d} G_h = C_d$. Since $C_d \cap B_{\hat{k}} = \emptyset$ holds, we have $v \notin B_{\hat{k}}$ in this case. Now, assume $I_v \neq \{1, \dots, d\}$. Let $i' \in \{1, \dots, d\} \setminus I_v$ be arbitrary. From $I_w \cup \{i\} \subseteq I_v$ we learn $i' \notin I_w$ and $i' \neq i$. Hence, we have $v, w \notin G_{i'}$. Since $w \notin B_{\hat{k}}$ holds (by the assumption at the outset of this claim's proof), we infer $\mu_{i'}(w) > \hat{k}$, i.e., the counter i' exceeds $\hat{k}$ in w before it falls below zero for the first time. Additionally, in v the counter i' falls below zero sometime. We have to show that it exceeds the value $\hat{k}$ before it first drops below zero.

Recall that i was the counter with the longest prefix w_1 of w with $\mu_i(w_1) \leq \hat{k}$. This implies $\mu_{i'}(w_1) > \hat{k}$. Note that w_1 is a prefix of $\mathrm{code}(\vec{u_1}) \dots \mathrm{code}(\vec{u_j})$ and therefore also of v. Hence, we have $\mu_{i'}(v) > \hat{k}$. Since i' was arbitrary, this holds for all counters in $\{1, \dots, d\} \setminus I_v$. In other words, for each $h \in \{1, \dots, d\}$ we have either $v \in G_h$ or $\mu_h(v) > \hat{k}$. Hence, $v \notin B_{\hat{k}}$ holds in this case. ◁

So, we have learned that there is another word $v \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$ having more non-negative counters I_v than w. Finally, induction yields a word $\hat{v} \in K(\mathfrak{V}_1, \mathfrak{V}_2) \setminus B_{\hat{k}}$ with $I_{\hat{v}} = \{1, \dots, d\}$, i.e., $\hat{v} \in \bigcap_{h=1}^{d} G_h = C_d$. This implies $\hat{v} \in C_d \cap K(\mathfrak{V}_1, \mathfrak{V}_2)$ - a contradiction to $K(\mathfrak{V}_1, \mathfrak{V}_2) \mid C_d$. ◀

With the help of Lemma 5.2 we are able to finally prove our main result Theorem 5.1.

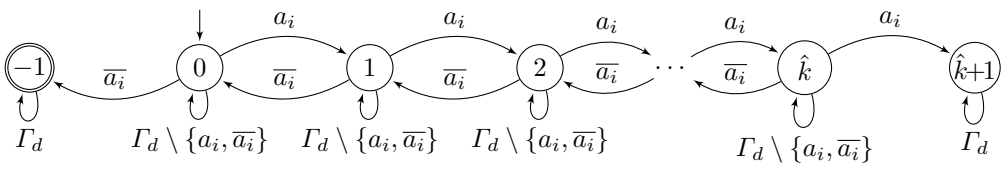


Figure 2 A DFA $\mathfrak{A}_i$ accepting the language $B_{\hat{k},i}$. It simulates the counter i bounded by $\hat{k}$.

Proof of Theorem 5.1. Since we have $K(\mathfrak{V}_1, \mathfrak{V}_2) \subseteq B_{\hat{k}}$ and $B_{\hat{k}} \cap C_d = \emptyset$, the set $B_{\hat{k}}$ is a separator of $K(\mathfrak{V}_1, \mathfrak{V}_2)$ and C_d. This language is also regular: in Figure 2 we depict a DFA $\mathfrak{A}_i$ accepting the language $B_{\hat{k},i} := \{w \in \Gamma_d^* \mid w \notin G_i \text{ and } \mu_i(w) \leq \hat{k}\}$ for $1 \leq i \leq d$. Since $B_{\hat{k}} = \bigcup_{i=1}^d B_{\hat{k},i}$ holds, we obtain a DFA accepting $B_{\hat{k}}$ using the classical product construction. The resulting DFA has the size $\prod_{i=1}^d |\mathfrak{A}_i| = \prod_{i=1}^d (\hat{k}+3) \in \mathcal{O}(\hat{k}^d)$.

We have seen that $B_{\hat{k}}$ separates $K(\mathfrak{V}_1, \mathfrak{V}_2)$ and C_d. With the same arguments as in Lemma 4.2, one shows that if R witnesses $T^{-1}(K) \mid L$, then $T(\Sigma^* \setminus R)$ witnesses $T(L) \mid K$. Since $K(\mathfrak{V}_1, \mathfrak{V}_2) = \mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1})^{-1}(\mathrm{L}(\mathfrak{V}_2))$ and $\mathrm{L}(\mathfrak{V}_1) = \mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1})(C_d)$, we conclude that

$$\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1})(\Gamma_d^* \setminus B_{\hat{k}}) \tag{2}$$

witnesses $\mathrm{L}(\mathfrak{V}_1) \mid \mathrm{L}(\mathfrak{V}_2)$. Since we have a DFA of size $\mathcal{O}(\hat{k}^d)$ for $B_{\hat{k}}$ and thus such a DFA for $\Gamma_d^* \setminus B_{\hat{k}}$, we obtain an NFA for (2) of size $\mathcal{O}(|Q_1| \cdot \hat{k}^d) = \mathcal{O}(|Q_1| \cdot \|\Delta_1\|^d \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)^d)$. ◀

The term $\mathcal{O}(|Q_1| \cdot \|\Delta_1\|^d \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)^d)$ from Theorem 5.1 is doubly exponential in d (and polynomial in the remaining numbers). In other words, for two given disjoint d-VASS $\mathfrak{V}_1$ and $\mathfrak{V}_2$ there is a doubly exponential sized NFA separating their languages $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$. If we are looking for a deterministic automaton separating these languages, we can use the power set construction to obtain a DFA of triply exponential size. The lower bounds by Czerwiński et. al. [10] show that these upper bounds are tight.

▶ **Corollary 5.5.** *From a given number $d \in \mathbb{N}^+$ and two disjoint d-VASS $\mathfrak{V}_1$ and $\mathfrak{V}_2$ we can compute*

(1) *an NFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ of size doubly exponential in d, $|\mathfrak{V}_1|$, and $|\mathfrak{V}_2|$.*

(2) *a DFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ of size triply exponential in d, $|\mathfrak{V}_1|$, and $|\mathfrak{V}_2|$.*

Proof.

(1) By Theorem 5.1 we can compute an NFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ with the following number of states:

$$\begin{aligned} &\mathcal{O}(|Q_1| \cdot \|\Delta_1\|^d \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)^d) \\ &= \mathcal{O}(|Q_1| \cdot \|\Delta_1\|^d \cdot (|Q_1| \cdot |Q_2| \cdot \max\{\|\Delta_1\|, \|\Delta_2\|\} + 2)^{((6d)!+1)\cdot d}), \end{aligned}$$

which is doubly exponential in d, $|\mathfrak{V}_1|$, and $|\mathfrak{V}_2|$.

(2) We can determinize the NFA from (1) using the classical power set construction. This results in an equivalent DFA of size exponential in the size of the NFA. ◀

Since the exponent of the term $\mathcal{O}(|Q_1| \cdot \|\Delta_1\|^d \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)^d)$ only depends on the dimension d, we could also ask for an upper bound of an NFA or DFA separating the languages of two VASS of fixed dimension. In this scenario we have to distinguish two cases: the numbers in our VASS are encoded in unary or binary. First, we consider the unary case. Here, we can construct a separating NFA of polynomial size and a DFA of exponential size.

▶ **Corollary 5.6.** *Fix a number $d \in \mathbb{N}^+$. From two disjoint d-VASS $\mathfrak{V}_1$ and $\mathfrak{V}_2$ in which the numbers are encoded in unary, we can compute*
(1) *an NFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ of size polynomial in $|\mathfrak{V}_1|$ and $|\mathfrak{V}_2|$.*
(2) *a DFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ of size exponential in $|\mathfrak{V}_1|$ and $|\mathfrak{V}_2|$.*

Proof.
(1) Since d is assumed to be fixed, the size of the regular separator of $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ from Theorem 5.1 is a polynomial in the sizes of $\mathfrak{V}_1$ and $\mathfrak{V}_2$.
(2) To achieve this result, we only have to determinize the NFA from statement (1). ◀

Now, we have to consider VASS of fixed dimension d with binary encoded numbers. To this end, we first have to introduce a binary norm: for a vector $\vec{v} \in \mathbb{Z}^d$ set $\|\vec{v}\|_2 := \log \|\vec{v}\|$. Based on this, we define the binary norm $\|\Delta\|_2$ of a set of transitions Δ. Slightly abusing terminology, when we speak of *VASS with binary encoding* (or *with binary encoded numbers*), then this only means we measure its size with $\|\cdot\|_2$ in place of $\|\cdot\|$. In this case, for two given VASS we find a separating NFA of exponential size and a separating DFA of doubly exponential size.

▶ **Corollary 5.7.** *Fix a number $d \in \mathbb{N}^+$. From two disjoint d-VASS $\mathfrak{V}_1$ and $\mathfrak{V}_2$ with binary encoding, we can compute*
(1) *an NFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ of size exponential in $|\mathfrak{V}_1|$ and $|\mathfrak{V}_2|$.*
(2) *a DFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ of size doubly exponential in $|\mathfrak{V}_1|$ and $|\mathfrak{V}_2|$.*

Proof.
(1) Since we encode numbers in binary the values $\|\Delta_1\|$ and $\|\Delta_2\|$ are exponential in the description size of $\mathfrak{V}_1$ resp. $\mathfrak{V}_2$. Hence, the NFA constructed in Theorem 5.1 has size exponential in the sizes of $\mathfrak{V}_1$ and $\mathfrak{V}_2$.
(2) Again, this is a direct consequence of the first statement using the classical power set construction to determinize the constructed NFA. ◀

5.1 Upper Bound for Binary Encoded 1-VASS

Interestingly, the given upper bound for a DFA separating the languages of two given binary encoded VASS of dimension 1 is not tight, yet. We can use a better construction than the classical power set construction to determinize our constructed separating NFA. In this case, we obtain a DFA which also has exponential size (in comparison to doubly exponential size with the power set construction).

▶ **Theorem 5.8.** *Given disjoint 1-VASS $\mathfrak{V}_1$ and $\mathfrak{V}_2$ with binary-encoded numbers, we can compute a DFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ of size exponential in $|\mathfrak{V}_1|$ and $|\mathfrak{V}_2|$.*

We take a closer look at the resulting NFA constructed in the last step of the proof of Theorem 5.1 (resp. in Corollary 5.7(1)). With the knowledge about this special NFA, we will apply an improved power set construction resulting in a DFA separating $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ without the exponential blowup.

So, let $\mathfrak{V}_i = (Q_i, \Delta_i, s_i, t_i)$ be two 1-VASS, $\mathfrak{T}_{\mathfrak{V}_1} = (Q_1, \delta_1, \{s_1\}, \{t_1\})$ be the rational transducer constructed from $\mathfrak{V}_1$ as described in the proof of Lemma 4.1 and let $\mathfrak{A} = (S, \Gamma_1, \delta_{\mathfrak{A}}, \{0\}, F_{\mathfrak{A}})$ be the DFA depicted in Figure 2 accepting $B_{\hat{k}}$ where $\hat{k} = \|\Delta_1\| \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2)$. In other words, we have $S = \{-1, 0, 1, \ldots, \hat{k}, \hat{k}+1\}$ and $F_{\mathfrak{A}} = \{-1\}$. The complement of $B_{\hat{k}}$ is accepted by the DFA $\overline{\mathfrak{A}} = (S, \Gamma_1, \delta_{\mathfrak{A}}, \{0\}, F_{\overline{\mathfrak{A}}})$ with $F_{\overline{\mathfrak{A}}} = \{0, 1, \ldots, \hat{k}+1\}$. In the following let $\leq$ denote the natural ordering on $S \subseteq \mathbb{Z}$. Then we can observe that $\overline{\mathfrak{A}}$'s transition relation $\delta_{\mathfrak{A}}$ is compatible with the ordering $\leq$:

▶ **Observation 5.9.** *Let $w \in \Gamma_1^*$ be a word and $m, m', n \in S$ with $m \xrightarrow{w}_{\overline{\mathfrak{A}}} n$ and $m' \geq m$. Then there is $n' \in S$ with $n' \geq n$ and $m' \xrightarrow{w}_{\overline{\mathfrak{A}}} n'$.*

In the next step we apply the rational transduction $\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1})$ to $\mathrm{L}(\overline{\mathfrak{A}}) = \Gamma_1^* \setminus B_{\hat{k}}$. We do this with the help of the classical construction resulting in the following NFA $\mathfrak{B} = (Q_{\mathfrak{B}}, \Sigma, \delta_{\mathfrak{B}}, I_{\mathfrak{B}}, F_{\mathfrak{B}})$ accepting $(\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1}))(\Gamma_1^* \setminus B_{\hat{k}})$:

- $Q_{\mathfrak{B}} = Q_1 \times S$ and $I_{\mathfrak{B}} = \{(s_1, 0)\}$,
- for all $(p, m), (q, n) \in Q_{\mathfrak{B}}$ and $a \in \Sigma$: $((p, m), a, (q, n)) \in \delta_{\mathfrak{B}}$ if, and only if, there is $w \in \Gamma_1^*$ with $p \xrightarrow{(w\,,\,a)}_{\mathfrak{T}_{\mathfrak{V}_1}} q$ and $m \xrightarrow{w}_{\overline{\mathfrak{A}}} n$, and
- for all $(p, m) \in Q_{\mathfrak{B}}$: $(p, m) \in F_{\mathfrak{B}}$ if, and only if, there are $w \in \Gamma_1^*$ and $n \in F_{\overline{\mathfrak{A}}}$ with $p \xrightarrow{(w\,,\,\varepsilon)}_{\mathfrak{T}_{\mathfrak{V}_1}} t_1$ and $m \xrightarrow{w}_{\overline{\mathfrak{A}}} n$.

Hence, $\mathfrak{B}$ is the separating NFA of exponential size from Corollary 5.7(1). We show next that the determinization of $\mathfrak{B}$ is possible without exponential blowup. To this end, we first need the following observation of $\mathfrak{B}$'s behavior namely that the compatibility of $\overline{\mathfrak{A}}$'s transition relation $\delta_{\mathfrak{A}}$ with $\leq$ is passed on to $\mathfrak{B}$'s transition relation $\delta_{\mathfrak{B}}$:

▶ **Lemma 5.10.** *Let $w \in \Sigma^*$, $(p, m), (q, n) \in Q_{\mathfrak{B}}$, and $m' \in S$ with $(p, m) \xrightarrow{w}_{\mathfrak{B}} (q, n)$ and $m' \geq m$. Then there is $n' \in S$ with $n' \geq n$ and $(p, m') \xrightarrow{w}_{\mathfrak{B}} (q, n')$.*

Proof. We prove this by induction on the length of w. If $w = \varepsilon$, then our statement is true: since $\mathfrak{B}$ has no ε-transitions, $(p, m) \xrightarrow{\varepsilon}_{\mathfrak{B}} (q, n)$ implies $p = q$ and $m = n$. Therefore, our statement holds for $n' = m'$.

Now, assume $w = w'a$ for some word $w' \in \Sigma^*$ and a letter $a \in \Sigma$. From $(p, m) \xrightarrow{w}_{\mathfrak{B}} (q, n)$ we learn that there is an intermediate state $(r, \ell) \in Q_{\mathfrak{B}}$ with $(p, m) \xrightarrow{w'}_{\mathfrak{B}} (r, \ell) \xrightarrow{a}_{\mathfrak{B}} (q, n)$. Since $|w'| < |w|$ holds, the induction hypothesis yields an $\ell' \in S$ with $\ell' \geq \ell$ and $(p, m') \xrightarrow{w'}_{\mathfrak{B}} (r, \ell')$. By the definition of the transition relation of $\mathfrak{B}$ we obtain from $(r, \ell) \xrightarrow{a}_{\mathfrak{B}} (q, n)$ a word $v \in \Gamma_1^*$ with $r \xrightarrow{(v,a)}_{\mathfrak{T}_{\mathfrak{V}_1}} q$ and $\ell \xrightarrow{v}_{\overline{\mathfrak{A}}} n$. According to Observation 5.9 there is $n' \in S$ with $n' \geq n$ and $\ell' \xrightarrow{v}_{\overline{\mathfrak{A}}} n'$. But this implies $(r, \ell') \xrightarrow{a}_{\mathfrak{B}} (q, n')$ and therefore $(p, m') \xrightarrow{w}_{\mathfrak{B}} (q, n')$. ◀

We can also show that the set of accepting states of $\mathfrak{B}$ is upwards closed wrt. the natural ordering of its set of states. This is the following lemma:

▶ **Lemma 5.11.** *Let $(p, m) \in F_{\mathfrak{B}}$ and $m' \in S$ with $m' \geq m$. Then we also have $(p, m') \in F_{\mathfrak{B}}$.*

Proof. By definition of $F_{\mathfrak{B}}$ there are $w \in \Gamma_1^*$ and $n \in F_{\overline{\mathfrak{A}}} = \{0, 1, \ldots, \hat{k}+1\}$ with $p \xrightarrow{(w,\varepsilon)}_{\mathfrak{T}_{\mathfrak{V}_1}} t_1$ and $m \xrightarrow{w}_{\overline{\mathfrak{A}}} n$. Due to Observation 5.9 there is $n' \in S$ with $n' \geq n$ and $m' \xrightarrow{w}_{\overline{\mathfrak{A}}} n'$. Since $n' \geq n \geq 0$ holds, we also learn $n' \in F_{\overline{\mathfrak{A}}}$ implying $(p, m') \in F_{\mathfrak{B}}$. ◀

Finally, we have to determinize the NFA $\mathfrak{B}$. To this end, we recall the classical power set construction of $\mathfrak{B}$: the result of this construction is the DFA $\mathfrak{P} = (2^{Q_{\mathfrak{B}}}, \Sigma, \delta_{\mathfrak{P}}, \{\iota_{\mathfrak{P}}\}, F_{\mathfrak{P}})$ where

- $\iota_{\mathfrak{P}} = \{(s_1, 0)\}$,
- $(X, a, Y) \in \delta_{\mathfrak{P}}$ if, and only if, $Y = \{y \in Q_{\mathfrak{B}} \mid \exists x \in X \colon (x, a, y) \in \delta_{\mathfrak{B}}\}$, and
- $F_{\mathfrak{P}} = \{X \subseteq Q_{\mathfrak{B}} \mid X \cap F_{\mathfrak{B}} \neq \emptyset\}$.

By induction we learn that $X \xrightarrow{w}_{\mathfrak{P}} Y$ holds if, and only if, Y is exactly the set of states that are reachable from X via w, i.e., $y \in Y$ iff there is $x \in X$ with $x \xrightarrow{w}_{\mathfrak{B}} y$. In particular, if Y is accepting and we have $\iota_{\mathfrak{P}} \xrightarrow{w}_{\mathfrak{P}} Y$, then there is $y \in Y \cap F_{\mathfrak{B}} \neq \emptyset$ with $(s_1, 0) \xrightarrow{w}_{\mathfrak{B}} y$. This means, an accepting run in $\mathfrak{P}$ also witnesses an accepting run in $\mathfrak{B}$.

Now, let $X \subseteq Q_{\mathfrak{B}}$ be some intermediate state of this w-labeled run from $\{(s_1, 0)\}$ to Y, i.e., we have $\iota_{\mathfrak{P}} \xrightarrow{u}_{\mathfrak{P}} X \xrightarrow{v}_{\mathfrak{P}} Y$ with $w = uv$. Let $(p, m) \in X$ and $(q, n) \in Y \cap F_{\mathfrak{B}}$ with $(s_1, 0) \xrightarrow{u}_{\mathfrak{B}} (p, m) \xrightarrow{v}_{\mathfrak{B}} (q, n)$. Assume that there is another state $(p, m') \in X$ with $m' \geq m$.

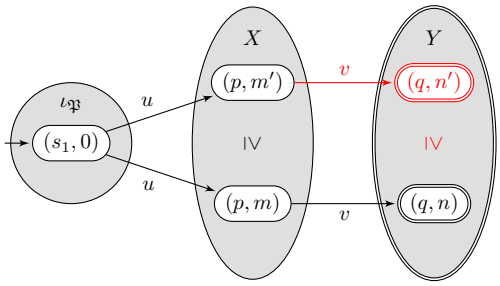


Figure 3 Visualization of the power set construction on the NFA $\mathfrak{B}$. The states in gray are states of the DFA $\mathfrak{P}$, the white ones are states of $\mathfrak{B}$. The reachability of and acceptance of (q, n') (red) is ensured by Lemmas 5.10 and 5.11.

Then Lemmas 5.10 and 5.11 state that there is also another state $(q, n') \in Y \cap F_{\mathfrak{B}}$ with $n' \geq n$ and $(s_1, 0) \xrightarrow{u}_{\mathfrak{B}} (p, m') \xrightarrow{v}_{\mathfrak{B}} (q, n')$, which also witnesses acceptance of w (cf. Figure 3, colored in red). This means, the set of w-labeled accepting runs of $\mathfrak{B}$ also is in some sense upwards closed. Therefore, it suffices to store only the greatest value $m \in S$ for each state $p \in Q_1$ such that $(p, m) \in X$ holds. This can be represented by a partial mapping from Q_1 into S. Here, we extend these partial mappings to maps with the help of a new symbol $\perp \notin S$, such that $f(q) = \perp$ means that f is undefined at q, that is, there is no such state (q, n). The result is a DFA having $(|S| + 1)^{|Q_1|}$ many states, which is exponential in the sizes of $\mathfrak{V}_1$ and $\mathfrak{V}_2$. This is much smaller than the size of $\mathfrak{P}$: it contains $2^{|Q_1| \cdot |S|}$ many states which is doubly exponential in $|\mathfrak{V}_1|$ and $|\mathfrak{V}_2|$.

Concretely, our DFA $\mathfrak{C} = (Q_{\mathfrak{C}}, \Sigma, \delta_{\mathfrak{C}}, \{\iota_{\mathfrak{C}}\}, F_{\mathfrak{C}})$ accepting $\mathrm{L}(\mathfrak{B}) = (\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1}))(\Gamma_1^* \setminus B_{\hat{k}})$ is defined as follows:

- $Q_{\mathfrak{C}} = (S \cup \{\perp\})^{Q_1}$, i.e., the set of all maps from Q_1 to $S \cup \{\perp\}$ with $\perp \notin S$; here, $f(q) = \perp$ means that no state $(q, n) \in Q_{\mathfrak{B}}$ is reachable,
- $\iota_{\mathfrak{C}} \colon Q_1 \to S \cup \{\perp\}$ with $\iota_{\mathfrak{C}}(s_1) = 0$ and $\iota_{\mathfrak{C}}(q) = \perp$ for each $q \in Q_1 \setminus \{s_1\}$,
- for all $f, g \in Q_{\mathfrak{C}}$ and $a \in \Sigma$: $(f, a, g) \in \delta_{\mathfrak{C}}$ if, and only if, for each $q \in Q_1$ we have $g(q) = \max\{n \in S \mid \exists p \in Q_1 \colon ((p, f(p)), a, (q, n)) \in \delta_{\mathfrak{B}}\}$ where $\max \emptyset := \perp$, and
- $F_{\mathfrak{C}} = \{f \in Q_{\mathfrak{C}} \mid \exists q \in Q_1 \colon (q, f(q)) \in F_{\mathfrak{B}}\}$.

We have to show now that our construction is correct, i.e., we show $\mathrm{L}(\mathfrak{C}) = \mathrm{L}(\mathfrak{B})$. We do this with the help of the following two propositions each proving one inclusion.

▶ **Proposition 5.12.** $\mathrm{L}(\mathfrak{C}) \subseteq \mathrm{L}(\mathfrak{B})$.

Proof. To prove this inclusion we first have to prove the following helping statement:

▷ Claim 5.13. Let $g \in Q_{\mathfrak{C}}$ and $w \in \Sigma^*$ with $\iota_{\mathfrak{C}} \xrightarrow{w}_{\mathfrak{C}} g$. Then we have $g(q) = \max\{n \in S \mid (s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, n)\}$ for each $q \in Q_1$ with $g(q) \neq \perp$.

Proof. We first show $g(q) \geq \max\{n \in S \mid (s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, n)\}$ for each $q \in Q_1$ with $g(q) \neq \perp$. We do this by induction on the length of w. So, if $w = \varepsilon$ the statement is obvious since $g = \iota_{\mathfrak{C}}$ holds in this case. Now, let $a \in \Sigma$ and $w' \in \Sigma^*$ with $w = w'a$. Then there is a state $f \in Q_{\mathfrak{C}}$ with $\iota_{\mathfrak{C}} \xrightarrow{w'}_{\mathfrak{C}} f \xrightarrow{a}_{\mathfrak{C}} g$.

Let $(p, m), (q, n) \in Q_{\mathfrak{B}}$ be arbitrary states with $(s_1, 0) \xrightarrow{w'}_{\mathfrak{B}} (p, m) \xrightarrow{a}_{\mathfrak{B}} (q, n)$. Since $|w'| < |w|$ holds, the induction hypothesis yields $f(p) \geq \max\{m' \in S \mid (s_1, 0) \xrightarrow{w'}_{\mathfrak{B}} (p, m')\}$ implying $f(p) \geq m$. From Lemma 5.10 we know that there is $n' \in S$ with $n' \geq n$ and $(p, f(p)) \xrightarrow{a}_{\mathfrak{B}} (q, n')$. Additionally, we know $(f, a, q) \in \delta_{\mathfrak{B}}$ implying $g(q) = \max\{n'' \in S \mid \exists p \in Q_1 \colon ((p, f(p)), a, (q, n'')) \in \delta_{\mathfrak{B}}\}$ and therefore $g(q) \geq n' \geq n$. Since $(q, n) \in Q_{\mathfrak{B}}$ was arbitrary, we infer $g(q) \geq \max\{n \in S \mid (s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, n)\}$.

Now we show the inverse inequality. To this end, we show $(s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, g(q))$ holds for all $q \in Q_1$ with $g(q) \neq \bot$. Again, we show this by induction on $|w|$. So, let $w = \varepsilon$. Then we have $g = \iota_{\mathfrak{C}}$ and $g(q) \neq \bot$ if, and only if, $q = s_1$. Obviously, we have $(s_1, 0) \xrightarrow{\varepsilon}_{\mathfrak{B}} (s_1, 0)$. Next, let $w = w'a$ for a letter $a \in \Sigma$ and a word $w' \in \Sigma^*$. There is $f \in Q_{\mathfrak{C}}$ with $\iota_{\mathfrak{C}} \xrightarrow{w'}_{\mathfrak{C}} f \xrightarrow{a}_{\mathfrak{C}} g$. The induction hypothesis yields $(s_1, 0) \xrightarrow{w'} (p, f(p))$ for each $p \in Q_1$ with $f(p) \neq \bot$. Let $q \in Q_1$ with $g(q) \neq \bot$. Due to $(f, a, g) \in \delta_{\mathfrak{C}}$ there is $p \in Q_1$ with $((p, f(p)), a, (q, g(q))) \in \delta_{\mathfrak{C}}$ (this also implies $f(p) \neq \bot$). Hence, we have $(s_1, 0) \xrightarrow{w'}_{\mathfrak{B}} (p, f(p)) \xrightarrow{a}_{\mathfrak{B}} (q, g(q))$. ◁

Let $w \in \mathrm{L}(\mathfrak{C})$. Then there is $g \in F_{\mathfrak{C}}$ with $\iota_{\mathfrak{C}} \xrightarrow{w}_{\mathfrak{C}} g$. Since g is accepting, there is a state $q \in Q_1$ with $(q, g(q)) \in F_{\mathfrak{B}}$. By definition of $\mathfrak{B}$ we know $g(q) \neq \bot$ in this case. By Claim 5.13 we know $g(q) = \max\{n \in S \mid (s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, n)\}$. Hence, we learn $(s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, g(q)) \in F_{\mathfrak{B}}$. This finally implies $w \in \mathrm{L}(\mathfrak{B})$. ◀

▶ **Proposition 5.14.** $\mathrm{L}(\mathfrak{B}) \subseteq \mathrm{L}(\mathfrak{C})$.

Proof. Again, we first need some helping statement:

▷ Claim 5.15. Let $w \in \Sigma^*$ and $(q, n) \in Q_{\mathfrak{B}}$ with $(s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, n)$. Then there is $g \in Q_{\mathfrak{C}}$ with $\iota_{\mathfrak{C}} \xrightarrow{w}_{\mathfrak{C}} g$ and $g(q) \geq n$.

Proof. We show this statement by induction on the length $|w|$ of the word w. The case $w = \varepsilon$ is obvious since $g := \iota_{\mathfrak{C}}$ satisfies $\iota_{\mathfrak{C}} \xrightarrow{\varepsilon}_{\mathfrak{C}} \iota_{\mathfrak{C}}$ and $\iota_{\mathfrak{C}}(s_1) = 0 \geq 0$ (note that $(q, n) = (s_1, 0)$ holds). Now, let $a \in \Sigma$ and $w' \in \Sigma^*$ with $w = w'a$. Then there is a state $(p, m) \in Q_{\mathfrak{B}}$ with $(s_1, 0) \xrightarrow{w'}_{\mathfrak{B}} (p, m) \xrightarrow{a}_{\mathfrak{B}} (q, n)$. By induction hypothesis there is $f \in Q_{\mathfrak{C}}$ with $\iota_{\mathfrak{C}} \xrightarrow{w}_{\mathfrak{C}} f$ and $f(p) \geq m$. According to Lemma 5.10 we also know that there is $n' \in S$ with $n' \geq n$ and $(p, f(p)) \xrightarrow{a}_{\mathfrak{B}} (q, n')$. In this case, we have $((p, f(p)), a, (q, n')) \in \delta_{\mathfrak{B}}$.

Let $g \in Q_{\mathfrak{C}}$ be the uniquely defined state with $(f, a, g) \in \delta_{\mathfrak{C}}$. Then we have $g(q)$ is the maximal value $n \in S$ with $((r, f(r)), a, (q, n)) \in \delta_{\mathfrak{B}}$ for some $r \in Q_1$. In particular, we have $g(q) \geq n' \geq n$ and $\iota_{\mathfrak{C}} \xrightarrow{w'}_{\mathfrak{C}} f \xrightarrow{a}_{\mathfrak{C}} g$. ◁

Now, let $w \in \mathrm{L}(\mathfrak{B})$. Then there is a state $(q, n) \in F_{\mathfrak{B}}$ with $(s_1, 0) \xrightarrow{w}_{\mathfrak{B}} (q, n)$. According to Claim 5.15 there is a state $g \in Q_{\mathfrak{C}}$ with $\iota_{\mathfrak{C}} \xrightarrow{w}_{\mathfrak{C}} g$ and $g(q) \geq n$. By Lemma 5.11 we also know $(q, g(q)) \in F_{\mathfrak{B}}$. But then we infer $g \in F_{\mathfrak{C}}$, i.e., $w \in \mathrm{L}(\mathfrak{C})$. ◀

Finally, we are able to prove the previously stated Theorem 5.8.

Proof of Theorem 5.8. The DFA $\mathfrak{C}$ as constructed above accepts the language $\mathrm{L}(\mathfrak{C}) = \mathrm{L}(\mathfrak{B}) = (\mathrm{R}(\mathfrak{T}_{\mathfrak{V}_1}))(\Gamma_1^* \setminus B_{\hat{k}})$ according to Propositions 5.12 and 5.14. This automaton has

$$\begin{aligned}|Q_{\mathfrak{C}}| &= |(S \cup \{\bot\})^{Q_1}| = |S \cup \{\bot\}|^{|Q_1|} \in \mathcal{O}(\hat{k}^{|Q_1|}) = \mathcal{O}((\|\Delta_1\| \cdot \mathrm{Rack}(\mathfrak{V}_1 \times \mathfrak{V}_2))^{|Q_1|}) \\ &= \mathcal{O}((\|\Delta_1\| \cdot (|Q_1| \cdot |Q_2| \cdot \max\{\|\Delta_1\|, \|\Delta_2\|\} + 2)^{6!+1})^{|Q_1|})\end{aligned}$$

many states. The last term is exponential in the size of $\mathfrak{V}_1$ and $\mathfrak{V}_2$. ◀

From the proof technique, we can extract a slightly more abstract statement that might be of independent interest. An *ordered NFA* is an NFA $\mathfrak{A}$ together with a quasi-ordering $(Q, \preccurlyeq)$ on its set of states Q such that if $p \preccurlyeq q$, then all words accepted from p are also accepted from q. An *anti-chain* of $\mathfrak{A}$ is an anti-chain in $(Q, \preccurlyeq)$.

▶ **Proposition 5.16.** *If $\mathfrak{A}$ is an ordered NFA whose anti-chains have at most ℓ states, then $\mathfrak{A}$ has an equivalent DFA with $|Q|^\ell$ states.*

Figure 4 1-VASS $\mathfrak{V}_n$ (left) and $\mathfrak{W}_n$ (right) in the proof of Proposition 6.3.

Here, the states of the DFA are the anti-chains of $(Q, \preccurlyeq)$. This is useful whenever ℓ is small (e.g. logarithmic) in the size of $\mathfrak{A}$ (i.e., in $|Q|$). In our proof, for example, one can equip $Q_{\mathfrak{B}} = Q_1 \times [-1, \hat{k}+1]$ with the ordering $(p, m) \preccurlyeq (q, n)$ if and only if $p = q$ and $m \leq n$. Then clearly, an anti-chain in $(Q_{\mathfrak{B}}, \preccurlyeq)$ contains at most $|Q_1|$ states.

6 Lower Bounds

In this final section we want to show that all of the upper bounds shown in Section 5 are tight. This means, whenever Corollaries 5.5–5.7 and Theorem 5.8 gives an i-fold exponential upper bound for separators with $i \geq 1$, then we shall here provide a sequence $(\mathfrak{V}_1, \mathfrak{W}_1), (\mathfrak{V}_2, \mathfrak{W}_2), \ldots$ of VASS $\mathfrak{V}_n, \mathfrak{W}_n$ of size polynomial in n such that any separator of $L(\mathfrak{V}_n)$ and $L(\mathfrak{W}_n)$ requires at least $\exp_i(n)$ states. Recall that $\exp_0(n) = n$ and $\exp_{i+1}(n) = 2^{\exp_i(n)}$ for $i \geq 0$.

The case where d is part of the input was already considered by Czerwiński et al. in [10] (they use the languages in Eq. (1) with $f\colon n \mapsto 2^{2^n}$). We mention this without proof:

▶ **Proposition 6.1** (Czerwiński et al. [10]). *For any $n \in \mathbb{N}$ there are disjoint VASS $\mathfrak{V}_n$ and $\mathfrak{W}_n$ of size polynomial in n such that*
(1) *any NFA separating $\mathrm{L}(\mathfrak{V}_n)$ and $\mathrm{L}(\mathfrak{W}_n)$ has at least 2^{2^n} states.*
(2) *any DFA separating $\mathrm{L}(\mathfrak{V}_n)$ and $\mathrm{L}(\mathfrak{W}_n)$ has at least $2^{2^{2^n}}$ states.*

This provides the lower bounds of the first row in Table 1. In the construction of Czerwiński et al., the dimension of $\mathfrak{V}_n$ and $\mathfrak{W}_n$ grows (polynomially) with n. This means, we need different constructions for fixed dimension. Moreover, in fixed dimension, we cannot translate between VASS with unary and binary encodings. This means, we have to distinguish between the two encodings. Let us begin with unary encodings (i.e. the blue entries in Table 1). In the case of NFA separators, our upper bounds are polynomial, so we need not prove any lower bounds. For DFA separators, the exponential lower bound is already achieved for VASS that have no counters:

▶ **Proposition 6.2.** *For any $n \in \mathbb{N}$ there are disjoint NFAs $\mathfrak{A}_n$ and $\mathfrak{B}_n$ of size polynomial in n such that any DFA separating $\mathrm{L}(\mathfrak{A}_n)$ and $\mathrm{L}(\mathfrak{B}_n)$ has at least 2^n states.*

Proof. For $n \in \mathbb{N}$ consider the languages $K_n = K_{f,n}$ and $L_n = L_{f,n}$ with $f\colon n \to n$, with $K_{f,n}, L_{f,n}$ as in Eq. (1). Both languages are regular and accepted by NFAs $\mathfrak{A}_n$ and $\mathfrak{B}_n$ with $\mathcal{O}(n)$ many states. Since we have $K_n = \{a, b\}^* \setminus L_n$, K_n is the only regular separator of K_n and L_n. But it is well-known that any DFA for K_n has at least 2^n states. ◀

This provides the lower bound for the two blue entries in Table 1. Let us now turn to binary encodings. For NFA separators, all lower bounds are achieved using 1-VASS: Our first proposition yields the lower bounds for the light and dark gray entries of Table 1.

▶ **Proposition 6.3.** *For any $n \in \mathbb{N}$, there are disjoint 1-VASS $\mathfrak{V}_n$ and $\mathfrak{W}_n$, with binary encoded numbers, of size polynomial in n such that any NFA (and thus any DFA) separating $\mathrm{L}(\mathfrak{V}_n)$ and $\mathrm{L}(\mathfrak{W}_n)$ has at least 2^n states.*

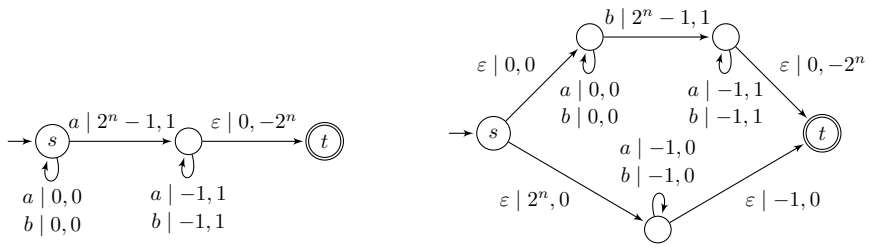


Figure 5 2-VASS $\mathfrak{V}_n$ (left) and $\mathfrak{W}_n$ (right) in the proof of Proposition 6.4.

Proof. Consider the languages $K_n = \{a^m \mid m < 2^n\}$ and $L_n = \{a^m \mid m \geq 2^n\}$. These two languages are accepted by the 1-VASS $\mathfrak{V}_n$ and $\mathfrak{W}_n$ as depicted in Figure 4. The transitions increasing resp. decreasing the counter by 2^n can be encoded in binary using n bits, i.e., $\mathfrak{V}_n$ and $\mathfrak{W}_n$ have size $\mathcal{O}(n)$.

Since K_n and L_n are regular and $K_n = \{a\}^* \setminus L_n$ holds, the only regular separator of K_n and L_n is K_n itself. It is easy to see that any NFA (and thus any DFA) accepting K_n requires at least 2^n many states. ◀

It remains to show the lower bound for the yellow entry of Table 1:

▶ **Proposition 6.4.** *For any $n \in \mathbb{N}$, there are disjoint 2-VASS $\mathfrak{V}_n$ and $\mathfrak{W}_n$, with binary encoded numbers, of size polynomial in n such that any DFA separating $\mathrm{L}(\mathfrak{V}_n)$ and $\mathrm{L}(\mathfrak{W}_n)$ has at least 2^{2^n} states.*

Proof. Consider the languages $K_n = K_{f,n}$ and $L_n = L_{f,n}$ with $f\colon n \mapsto 2^n$ with $K_{f,n}, L_{f,n}$ as in Eq. (1). These two languages are accepted by the 2-VASS $\mathfrak{V}_n$ and $\mathfrak{W}_n$, resp., as depicted in Figure 5. It is clear that both VASS have size $\mathcal{O}(n)$. Similar to the proof of Proposition 6.2 we can see that any DFA accepting K_n has at least 2^{2^n} many states. ◀

References

1. Pablo Barceló, Diego Figueira, and Rémi Morvan. Separating automatic relations. *CoRR*, abs/2305.08727, 2023. doi:10.48550/arXiv.2305.08727.
2. Pascal Baumann, Roland Meyer, and Georg Zetzsche. Regular separability in Büchi VASS. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.9.
3. Mikołaj Bojańczyk. It is undecidable if two regular tree languages can be separated by a deterministic tree-walking automaton. *Fundam. Informaticae*, 154(1-4):37–46, 2017. doi:10.3233/FI-2017-1551.
4. Laura Bozzelli and Pierre Ganty. Complexity Analysis of the Backward Coverability Algorithm for VASS. In Giorgio Delzanno and Igor Potapov, editors, *Reachability Problems*, Lecture Notes in Computer Science, pages 96–109, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-24288-5_10.
5. Christian Choffrut and Serge Grigorieff. Separability of rational relations in $A^* \times \mathbb{N}^m$ by recognizable relations is decidable. *Inf. Process. Lett.*, 99(1):27–32, 2006. doi:10.1016/j.ipl.2005.09.018.
6. Lorenzo Clemente, Wojciech Czerwiński, Slawomir Lasota, and Charles Paperman. Regular Separability of Parikh Automata. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 117:1–117:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.117.

7 Lorenzo Clemente, Wojciech Czerwiński, Slawomir Lasota, and Charles Paperman. Separability of Reachability Sets of Vector Addition Systems. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 24:1–24:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.24.

8 Lorenzo Clemente and Michał Skrzypczak. Deterministic and game separability for regular languages of infinite trees. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 126:1–126:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.126.

9 Wojciech Czerwiński and Slawomir Lasota. Regular separability of one counter automata. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005079.

10 Wojciech Czerwiński, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory (CONCUR 2018)*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CONCUR.2018.35.

11 Wojciech Czerwiński, Wim Martens, and Tomás Masopust. Efficient Separability of Regular Languages by Subsequences and Suffixes. 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 II*, volume 7966 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2013. doi:10.1007/978-3-642-39212-2_16.

12 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, Marc Zeitoun, and Georg Zetzsche. A Characterization for Decidable Separability by Piecewise Testable Languages. *Discrete Mathematics and Theoretical Computer Science*, 19(4), 2017. doi:10.23638/DMTCS-19-4-1.

13 Wojciech Czerwiński and Georg Zetzsche. An Approach to Regular Separability in Vector Addition Systems. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 341–354, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3373718.3394776.

14 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.

15 Jean Goubault-Larrecq and Sylvain Schmitz. Deciding Piecewise Testable Separability for Regular Tree Languages. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 97:1–97:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.97.

16 Sheila A. Greibach. Remarks on Blind and Partially Blind One-Way Multicounter Machines. *Theoretical Computer Science*, 7(3):311–324, 1978. doi:10.1016/0304-3975(78)90020-8.

17 Christopher Hugenroth. Separating Regular Languages over Infinite Words with Respect to the Wagner Hierarchy. In Mikołaj Bojańczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021)*, volume 213 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:13, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2021.46.

18 Harry B. Hunt III. On the Decidability of Grammar Problems. *Journal of the ACM*, 29(2):429–447, 1982. doi:10.1145/322307.322317.

19 Matthias Jantzen. On the hierarchy of Petri net languages. *RAIRO – Theoretical Informatics and Applications*, 13(1):19–30, 1979. doi:10.1051/ita/1979130100191.

20 Eren Keskin and Roland Meyer. Separability and non-determinizability of WSTS. *CoRR*, abs/2305.02736, 2023. doi:10.48550/arXiv.2305.02736.

21 Eryk Kopczyński. Invisible Pushdown Languages. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 867–872. ACM, 2016. doi:10.1145/2933575.2933579.

22 Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Węgrzycki. Coverability in VASS revisited: Improving rackoff's bound to obtain conditional optimality. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 131:1–131:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.131.

23 Tomás Masopust. Separability by piecewise testable languages is PTime-complete. *Theor. Comput. Sci.*, 711:109–114, 2018. doi:10.1016/j.tcs.2017.11.004.

24 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating Regular Languages by Locally Testable and Locally Threshold Testable Languages. In Anil Seth and Nisheeth K. Vishnoi, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, volume 24 of *LIPIcs*, pages 363–375. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.FSTTCS.2013.363.

25 Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. 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 75:1–75:10. ACM, 2014. doi:10.1145/2603088.2603098.

26 Thomas Place and Marc Zeitoun. Separation and the Successor Relation. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 662–675. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.662.

27 Thomas Place and Marc Zeitoun. Separating Regular Languages with First-Order Logic. *Log. Methods Comput. Sci.*, 12(1), 2016. doi:10.2168/LMCS-12(1:5)2016.

28 Thomas Place and Marc Zeitoun. Separating Without Any Ambiguity. 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 137:1–137:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.137.

29 Thomas Place and Marc Zeitoun. Separation and covering for group based concatenation hierarchies. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785655.

30 Thomas Place and Marc Zeitoun. Separation for dot-depth two. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/lmcs-17(3:24)2021.

31 Thomas Place and Marc Zeitoun. A generic polynomial time approach to separation by first-order logic without quantifier alternation. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 43:1–43:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.43.

32 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

33 Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2), 1976.

34 Ramanathan S. Thinniyam and Georg Zetzsche. Regular separability and intersection emptiness are independent problems. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 51:1–51:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.51.

35 Georg Zetzsche. Separability by piecewise testable languages and downward closures beyond subwords. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 929–938. ACM, 2018. doi:10.1145/3209108.3209201.

Acyclic Petri and Workflow Nets with Resets

Dmitry Chistikov
Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, Coventry, UK

Wojciech Czerwiński
University of Warsaw, Poland

Piotr Hofman
University of Warsaw, Poland

Filip Mazowiecki
University of Warsaw, Poland

Henry Sinclair-Banks
Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, Coventry, UK

Abstract

In this paper we propose two new subclasses of Petri nets with resets, for which the reachability and coverability problems become tractable. Namely, we add an acyclicity condition that only applies to the consumptions and productions, not the resets. The first class is acyclic Petri nets with resets, and we show that coverability is PSPACE-complete for them. This contrasts the known Ackermann-hardness for coverability in (not necessarily acyclic) Petri nets with resets. We prove that the reachability problem remains undecidable for acyclic Petri nets with resets. The second class concerns workflow nets, a practically motivated and natural subclass of Petri nets. Here, we show that both coverability and reachability in acyclic workflow nets with resets are PSPACE-complete. Without the acyclicity condition, reachability and coverability in workflow nets with resets are known to be equally hard as for Petri nets with resets, that being Ackermann-hard and undecidable, respectively.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Petri nets, Workflow Nets, Resets, Acyclic, Reachability, Coverability

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.16

Related Version *Full Version*: `https://arxiv.org/abs/2310.01992`

Funding *Dmitry Chistikov*: Supported in part by the Engineering and Physical Sciences Research Council [EP/X03027X/1].
Wojciech Czerwiński: Supported by the ERC grant INFSYS, agreement no. 950398.
Piotr Hofman: Supported by the ERC grant INFSYS, agreement no. 950398.
Filip Mazowiecki: Supported by the ERC grant INFSYS, agreement no. 950398.
Henry Sinclair-Banks: Supported by EPSRC Standard Research Studentship (DTP), grant number EP/T5179X/1.

Acknowledgements We would like to thank Alain Finkel for pointing out an error in the introduction. We would also like to thank our anonymous reviewers for their detailed comments.

1 Introduction

Petri nets [22] are among the most fundamental formalisms for modelling processes. They are defined by a finite set of places and a finite set of transitions. A configuration of a Petri net, known as a marking, is a vector of dimension equal to the number of places, with entries equal to the number of tokens in particular places. Transitions change markings by consuming and producing tokens in places. For an example, see Figure 1.

© Dmitry Chistikov, Wojciech Czerwiński, Piotr Hofman, Filip Mazowiecki, and Henry Sinclair-Banks; licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 16; pp. 16:1–16:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

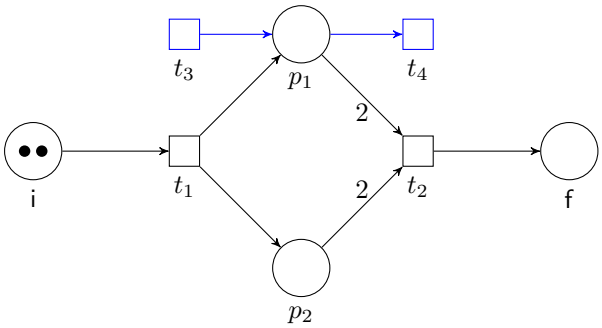


Figure 1 An example Petri net with four places i, p_1, p_2, f and four transitions t_1, t_2, t_3, t_4. Arcs pointing to transitions consume tokens from the respective places, and arcs pointing away from transitions produce tokens in the respective places. Arcs without labels denote single token consumption or production. Other labels, such as "2" in this example, are explicit. Initially, the marking can be represented by the vector $(2, 0, 0, 0)$ where there are two tokens in i, and no tokens in the other three places. At this marking, the transition t_2 cannot be fired as it needs to consume 2 tokens from each of p_1 and p_2. One can see that by firing a sequence of transitions (t_1, t_1, t_2) we reach the marking $(0, 0, 0, 1)$. The transitions t_3 and t_4 are highlighted (in blue) because t_3 does not consume any tokens and t_4 does not produce any tokens.

The central decision problem for Petri nets is the *reachability problem*. Given a Petri net, an initial marking, and a target marking, reachability asks whether there is a run between the two markings. Reachability in Petri nets is a decision problem with non-primitive recursive complexity [17, 18], recently it has been shown to be Ackermann-complete [7, 8]. The *coverability problem*, a relaxation of the reachability problem, asks whether there is a run that reaches a marking at least as great as the target marking. Coverability is provably simpler than reachability and it is known to be EXPSPACE-complete [20, 23, 16]. In the example (Figure 1), from the initial marking $(2, 0, 0, 0)$, one can reach $(0, 0, 0, 1)$, but one cannot reach $(0, 0, 1, 0)$. However, $(0, 0, 1, 0)$ can be covered since $(1, 1, 1, 0)$ can be reached.

In this paper, we consider Petri nets that are equipped with *resets* but are restricted to be *acyclic*. Resets are an extra feature of transitions that allow transitions to empty a subset of places. In modelling processes, resets offer the ability to express cancellation, which is important in many applications [29, Table 1]. Unfortunately, in general without any acyclicity restriction, for Petri nets with resets, reachability is undecidable [1, 9] and coverability is Ackermann-complete [24, 11]. Therefore, in order to observe the decidability of the reachability problem one needs to focus on a subclass of Petri nets with resets. A natural restriction is *acyclicity* that applies to the graph representation of the Petri net. For example, observe that the Petri net in Figure 1 is acyclic since the arcs do not induce any cycles between the places and transitions. Both reachability and coverability in acyclic Petri nets are NP-complete [19]. The NP upper bound is straightforward: it suffices to guess how many times each transition is fired in the run. It is always possible to transform this guess into an actual run by sorting the transitions in a topological order induced by the acyclic structure. As far as we know, acyclic Petri nets with resets have not been studied previously; they are a natural candidate for an expressive yet tractable class of Petri nets. We remark that the NP upper bound argument for reachability does not translate to the model with resets; changing the order of the resets does not preserve the reached marking.

We study Petri nets and their popular subclass *workflow nets* [28]. Workflow nets are Petri nets that have two special places, an input place i and an output place f. The places and transitions are also restricted so that no tokens can be produced in i, no tokens can

	Coverability	Reachability
Acyclic workflow nets with resets	PSPACE-complete (Section 4.1)	PSPACE-complete (Section 3.1)
Acyclic Petri nets with resets	PSPACE-complete (Section 3.2)	Undecidable (Section 4.2)

Figure 2 A summary of our results. Section 4.1 contains the PSPACE lower bound and Section 3.1 and Section 3.2 contain the PSPACE upper bounds.

be consumed from f, and all places and transitions lie on paths from i to f. The Petri net in Figure 1 without the (blue) highlighted transitions, t_3 and t_4, is a workflow net. Many practical instances of Petri nets are workflow nets [10], which forbid unnatural behaviour. Workflow nets are well studied [30], also with resets [29]. The complexities of the reachability and coverability problems for workflow nets are the same as for Petri nets. Indeed, the special places i and f produce and consume the initial and target markings, respectively. By introducing additional "artificial" places, it is not challenging to ensure that all places and transitions are on some path from i to f. However, the last construction does not preserve acyclicity. It turns out that acyclic Petri nets are more involved than acyclic workflow nets. For example, while the set of markings reachable from the initial marking is always finite for acyclic workflow nets [25], this is not true for acyclic Petri nets. In Figure 1, place p_1 can contain arbitrarily many tokens by firing t_3. In contrast, the workflow net obtained by removing transitions t_3 and t_4 will never contain more than 2 tokens in any place.

Our results. We determine the complexity of reachability and coverability in both acyclic Petri nets with resets and acyclic workflow nets with resets. We prove that coverability in acyclic Petri nets with resets is PSPACE-complete. Further, we show that both reachability and coverability in acyclic workflow nets with resets are also PSPACE-complete. On the other hand, we prove that, rather surprisingly, reachability in acyclic Petri nets with resets is undecidable. A summary of our results is in Figure 2.

For reachability in acyclic workflow nets with resets, we argue that a place cannot contain more than an exponential number of tokens with respect to the size of the reachability instance. The proof is comparable to the proof of the NP upper bound for acyclic Petri nets: one can reorder the firing sequence of transitions according to a topological order induced by acyclicity.

▶ **Theorem 3.1.** *Reachability in acyclic workflow nets with resets is in* PSPACE.

For coverability in acyclic Petri nets with resets, we show that there are two cases for the number of tokens that a place may contain. A place may either take at most an exponential number M of tokens, or it can take an arbitrarily large number of tokens, represented by ω. By abstracting the space of markings to a subset of $\{0, 1, \ldots, M-1, M, \omega\}^n$, we can search for a coverability run in polynomial space.

▶ **Theorem 3.2.** *Coverability in acyclic Petri nets with resets is in* PSPACE.

We complement these upper bounds with matching lower bounds. We show that coverability in acyclic workflow nets with resets requires polynomial space via a polynomial time reduction from QSAT. In the reduction, we construct an acyclic workflow net with resets that simulates assignments to the quantified variables (using places whose non-emptiness corresponds to the satisfaction of a literal) and checks that the formula evaluates to true for each assignment (using places whose non-emptiness corresponds to the satisfaction of a clause).

▶ **Theorem 4.2.** *Coverability in acyclic workflow nets with resets is* PSPACE*-hard.*

These three results allow us to conclude that coverability in acyclic Petri nets with resets and both coverability and reachability in acyclic workflow nets with resets are PSPACE-complete problems. We contrast this with the undecidability of reachability in acyclic Petri net with resets. Our proof is a reduction from reachability in general Petri nets with resets, which is known to be undecidable [1]. The core of our proof is the ability to simulate transitions whose arcs are not acyclic.

▶ **Theorem 4.13.** *Reachability in acyclic Petri nets with resets is undecidable.*

Related work. For workflow nets, a central decision problem is the *soundness* problem. An instance of soundness usually fixes the initial and target markings to only have one token in i and f, respectively. The soundness problem asks whether every marking reachable from the initial marking can then go on to reach the target marking. For workflow nets, it is known that soundness reduces to reachability [28], and an optimal algorithm for soundness (which does not rely on reachability) was only recently presented [3]. Variants of reachability and coverability have also been used as relaxations to implement soundness [26, 4]. Thus, we expect this work to provide a good background to study soundness on acyclic workflow nets with resets in the future.

In order to obtain decidability for the reachability problem on Petri nets with resets we both restrict the class to workflow nets and enforce acyclicity. However, instead of relaxing the class of Petri nets, one could allow the places to contain a negative number of tokens. Reachability in this relaxed model is called *integer reachability* and is known to be in NP for Petri nets[1], even with resets [6].

There are many extensions of Petri nets other than adding resets. We would like to highlight one extension in particular: Petri nets with transfers. Similar to resets, transfers move all the tokens from one place to another (instead of just removing them) [1]. Transfers allow the modelling of some properties of C programs [15]. For Petri nets with transfers, reachability in undecidable [1], but, coverability is decidable [9]. More generally, Petri nets with transfers and Petri nets with resets are examples of affine Petri nets [27, 12]. The previously mentioned integer reachability problem has been studied for this broad class of Petri nets [2, 5]. Consequently, integer reachability in Petri nets with transfers is in PSPACE [2]. As far as we know, reachability and coverability have not been considered for acyclic Petri nets with transfers or acyclic affine Petri nets, which we leave as possible future work.

2 Preliminaries

Let $\mathbb{Z}$ be the set of *integers* and $\mathbb{N}$ the set of *natural numbers* (nonnegative integers). Let ω stand for the first infinite cardinal, i.e. $\omega = |\mathbb{N}|$. Symbols $\mathbb{Z}_\omega$ and $\mathbb{N}_\omega$ denote the set of natural numbers and the set of integer numbers, each extended with ω, respectively. As usual, $|S|$ denotes the number of elements of a set S. We denote intervals by $[x, y] = \{z \in \mathbb{Z} \mid x \leq z \leq y\}$.

We use boldface to denote vectors, and we specify a vector by listing its coordinates, which are indexed using square brackets, in a tuple, so $\mathbf{v} = (\mathbf{v}[1], \ldots, \mathbf{v}[k])$. For two vectors $\mathbf{v}$ and $\mathbf{w}$ of equal dimension, we write $\mathbf{v} \geq \mathbf{w}$ if for every coordinate s we have $\mathbf{v}[s] \geq \mathbf{w}[s]$. If $\mathbf{v} \geq \mathbf{w}$ and $\mathbf{v} \neq \mathbf{w}$, then $\mathbf{v} > \mathbf{w}$; this partial order is called the pointwise order of vectors.

[1] Integer reachability is NP-complete for vector addition systems with states [13].

A vector $\mathbf{v}$ is non-negative if $\mathbf{v} \geq (0, 0, \ldots, 0)$. The norm $\|\cdot\|$ of a k-dimensional vector $\mathbf{v}$ is the sum of absolute values of its coordinates that are not equal to ω: $\|\mathbf{v}\| = \sum_{\mathbf{v}[i]\in\mathbb{N}} |\mathbf{v}[i]|$. We overload notation by saying that the norm $\|\cdot\|$ of a collection of vectors V is the sum of the norms of vectors in V, $\|V\| = \sum_{\mathbf{v}\in V} \|\mathbf{v}\|$.

Petri nets. A *Petri net* is a tuple (P, T, F) consisting of a finite set of *places* P, a finite set of *transitions* T (disjoint from P), and a function defining the *arcs* $F\colon (P\times T)\cup(T\times P) \to \mathbb{N}$. There is an arc from x to y for $(x, y) \in (P\times T)\cup(T\times P)$ if and only if $F(x, y) > 0$. In diagrams, this arc is *labelled* with the value of $F(x, y)$. One can view Petri nets as labelled graphs where $P\cup T$ is the set of nodes, and arcs are edges, labelled according to F. For example, in Figure 1 for transition t_1 we have $F(\mathrm{i}, t_1) = F(t_1, p_1) = F(t_1, p_2) = 1$ and all other values involving t_1 are 0. We can define a *path* in a Petri net as a sequence of places and transitions connected by arcs. A Petri net is *acyclic* if the graph of places and transitions with arcs is acyclic. The *norm* of a Petri net $\mathcal{N} = (P, T, F)$ is $\|\mathcal{N}\| = |P| \cdot |T| + \sum_{p\in P, t\in T}(F(t, p) + F(p, t))$.

▶ **Definition 2.1.** *A* Petri net with resets *is a tuple* (P, T, F, R)*, where* (P, T, F) *is a Petri net and* $R\colon T \to 2^P$ *is a function defining* reset edges*. There is reset edge between a transition* $t \in T$ *and a place* $p \in P$ *if and only if* $p \in R(t)$*. A Petri net with resets* (P, T, F, R) *is an* acyclic Petri net with resets *if* (P, T, F) *is acyclic according to the definition above.*

Importantly, reset edges are *not* subject to the acyclicity restriction. We discuss this in more detail below, after the formal definition of the semantics.

The norm of a Petri net with resets is $\|(P, T, F, R)\| = \|(P, T, F)\| + \sum_{t\in T} |R(t)|$.

For a Petri net with resets (P, T, F, R), the *pre-vector* of a transition t is ${}^\bullet t\colon P \to \mathbb{N}$, where ${}^\bullet t[p] = F(p, t)$, and its *post-vector* is $t^\bullet\colon P \to \mathbb{N}$, where $t^\bullet[q] = F(t, q)$. We use similar notation for the *reset-operator* $t^\circ \subseteq P$, namely $t^\circ = R(t)$.

Let us define the semantics of Petri nets (with resets). The collection of *markings* of a Petri net with resets (P, T, F, R) is the set of all vectors in $\mathbb{N}^P$. Places are said to contain *tokens*, a finite resource that can be *consumed*, *produced*, and *reset* by transitions. For a given marking $\mathbf{m}$, a place p contains tokens if $\mathbf{m}[p] > 0$, otherwise it is *empty*. A transition t can be *fired* at a marking $\mathbf{m}$ if and only if $\mathbf{m} \geq {}^\bullet t$. The firing proceeds through the following phases (see Figure 3 for an example):

- first, tokens are *consumed*, which results in $\mathbf{m}' = \mathbf{m} - {}^\bullet t$;
- then, places are *reset*, which results in $\mathbf{m}''$ where $\mathbf{m}''[p] = 0$ for all $p \in t^\circ$ and $\mathbf{m}''[p] = \mathbf{m}'[p]$ for all $p \notin t^\circ$;
- finally, tokens are *produced*, which results in the new marking $\mathbf{n} = \mathbf{m}'' + t^\bullet$.

We write $\mathbf{m} \xrightarrow{t} \mathbf{n}$.

▶ Note. In the semantics of Petri nets with resets, whether or not a place under reset contains tokens does not affect whether the transition can be fired. This makes the effect of resets distinct from the usual consumption of tokens by a transition. Resets do not produce any tokens either. Thus, resets are considered "undirected", and we refer to reset *edges* (rather than arcs). For the sake of clarity, in all drawings of Petri nets with resets, the reset edges are undirected and will be coloured red to distinguish them further.

A *firing sequence* $\sigma = (t_1, t_2, \ldots, t_n)$ is a sequence of transitions. It forms a *run* from a marking $\mathbf{m}_0$ to a marking $\mathbf{m}_n$ if $\mathbf{m}_0 \xrightarrow{t_1} \mathbf{m}_1 \xrightarrow{t_2} \mathbf{m}_2 \xrightarrow{t_3} \cdots \xrightarrow{t_n} \mathbf{m}_n$ for some intermediate markings $\mathbf{m}_1, \ldots, \mathbf{m}_{n-1}$. The run is denoted $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}_n$. We also write $\mathbf{m} \xrightarrow{*} \mathbf{n}$ if there exists a run from $\mathbf{m}$ to $\mathbf{n}$; in this case we say that $\mathbf{n}$ is *reachable* from $\mathbf{m}$. Further, we say that a run $\mathbf{m} \xrightarrow{*} \mathbf{n}'$ *covers* $\mathbf{n}$ if $\mathbf{n}' \geq \mathbf{n}$. If such a σ exists. we say that $\mathbf{n}$ can be *covered* from $\mathbf{m}$.

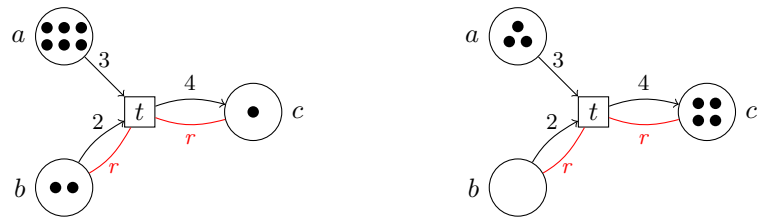


Figure 3 Two markings of an acyclic Petri net with resets with three places a, b, and c. Left: upon firing t from marking $(6, 2, 1)$ shown, 3 tokens are consumed from a and 2 tokens are consumed from b. Then, b and c are reset to 0 tokens, from 0 tokens and 1 token, respectively. Finally, 4 tokens are produced in c; this is the only number of tokens c can contain after t is fired. Right: marking $(3, 0, 4)$ is reached as the result of firing t.

▶ Note. Every Petri net can be seen as a Petri net with resets whose reset function is null, $R(t) = \emptyset$ for all t. So all definitions for Petri nets with resets naturally extend to Petri nets.

Workflow nets. A *workflow net* is a triple $(\mathcal{P}, \mathsf{i}, \mathsf{f})$ where $\mathcal{P}$ is a Petri net (P, T, F), $\mathsf{i} \in P$ is the *initial place*, $\mathsf{f} \in P$ is the *final place*, and all places and transitions lie on paths from i to f. A *workflow net with resets* is a triple $(\mathcal{R}, \mathsf{i}, \mathsf{f})$ where $\mathcal{R} = (P, T, F, R)$ is a Petri net with resets, and $((P, T, F), \mathsf{i}, \mathsf{f})$ is a workflow net. We say that a workflow net (with resets) $(\mathcal{N}, \mathsf{i}, \mathsf{f})$ is *acyclic* if the Petri net (with resets) $\mathcal{N}$ is acyclic. In Figure 1 the Petri net without transitions t_3 and t_4 is also a workflow net.

Decision problems. The following problems can be posed with any combination of added resets, acyclicity, and the workflow restriction.

Reachability in Petri nets

INPUT: A Petri net $\mathcal{N}$, an initial marking $\mathbf{m}$, and a target marking $\mathbf{n}$.
QUESTION: Does there exist a firing sequence σ such that $\mathbf{m} \xrightarrow{\sigma} \mathbf{n}$?

Coverability in Petri nets

INPUT: A Petri net $\mathcal{N}$, an initial marking $\mathbf{m}$, and a target marking $\mathbf{n}$.
QUESTION: Does there exist a firing sequence σ such that $\mathbf{m} \xrightarrow{\sigma} \mathbf{n}'$, where $\mathbf{n}' \geq \mathbf{n}$?

To give *instances* of these problems, we use tuples $(\mathcal{N}, \mathbf{m}, \mathbf{n})$. The *norm* of an instance $\|(\mathcal{N}, \mathbf{m}, \mathbf{n})\| = \|\mathcal{N}\| + \|\mathbf{m}\| + \|\mathbf{n}\|$. Depending on whether the arc weights are written in unary or binary, the *bit size* of the input is polynomial in the norm or logarithmic in the norm, respectively. Unary encoding suffices for our PSPACE lower bound (Theorem 4.2); see Lemma 4.1. Both of our PSPACE upper bounds (Theorem 3.1 and Theorem 3.2) hold even when the arc weights are binary-encoded. The undecidability result (Theorem 4.13) is independent of the encoding.

3 Upper Bounds

3.1 Reachability in Acyclic Workflow Nets with Resets

▶ **Theorem 3.1.** *Reachability in acyclic workflow nets with resets is in* PSPACE.

Proof. We rely on the simple property that reachable markings in acyclic workflow nets with resets are exponentially bounded. Let $\mathcal{R} = (P, T, F, R)$ be a given acyclic workflow net with resets and fix an initial marking $\mathbf{m}$. Consider the workflow net $\mathcal{W} = (P, T, F)$ that is just

$\mathcal{R}$ with the resets removed. Suppose from a marking $\mathbf{p}$ in $\mathcal{R}$, firing a transition t leads to marking $\mathbf{q}$. Clearly with the resets removed, firing t from $\mathbf{p}$ in $\mathcal{W}$ leads to a marking $\mathbf{q}'$ and $\mathbf{q}' \geq \mathbf{q}$. Notice also that the removal of resets does not alter whether or not a transition can be fired, if a transition can be fired from $\mathbf{p}$ then it can be fired from any $\mathbf{p}' \geq \mathbf{p}$. It follows that if $\mathbf{m} \xrightarrow{\pi} \mathbf{n}$ in $\mathcal{R}$, then $\mathbf{m} \xrightarrow{\pi} \mathbf{n}'$ in $\mathcal{W}$ for some $\mathbf{n}' \geq \mathbf{n}$. With this in mind, it suffices to argue that any reachable marking in $\mathcal{W}$ can be stored in polynomial space, relative to the norms of $\mathbf{m}$ and $\mathcal{W}$.

Let $m = \|\mathcal{R}\| + \|\mathbf{m}\|$. We prove that if $\mathbf{m} \xrightarrow{\pi} \mathbf{n}'$ in $\mathcal{W}$, then $\|\mathbf{n}'\| \leq m^{n+1}$, where n is the number of distinct transitions occurring in the firing sequence π. Since $\mathcal{W}$ is acyclic, there is a topological order on (the sources of) the transitions, and π can be permuted to respect this order (cf. [14]). Every transition in a workflow net must consume at least one token, so it follows that the i-th distinct transition can be fired at most m^i many times, resulting in a marking of norm at most m^{i+1}. Therefore, the norm of the largest possible marking is m^{n+1} and since $n \leq |T|$, all markings observed in the (permuted) run can be written down using polynomially many bits. Hence, reachability in acyclic workflow nets with resets can be decided using polynomial space. ◀

3.2 Coverability in Acyclic Petri Nets with Resets

▶ **Theorem 3.2.** *Coverability in acyclic Petri nets with resets is in* PSPACE.

We fix our attention on an instance $(\mathcal{P}, \mathbf{m}, \mathbf{n})$ of coverability in acyclic Petri nets with resets. Our approach can be summarised in two parts. First, we construct another infinite-state system $\mathcal{N}$ by modifying $\mathcal{P}$, that is much like a Petri net. The difference is that the places of $\mathcal{N}$ may contain an "infinite" number of tokens, denoted ω. Importantly, we will argue that $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{P}$ if and only if $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{N}$. Second, we show that the set of markings reachable from $\mathbf{m}$ in $\mathcal{N}$ has cardinality exponential in $\|\mathcal{N}\|$ and $\|\mathbf{m}\|$. Together, this allows us to decide, in polynomial space, this instance of coverability in acyclic Petri nets with resets.

We say that a transition t is *generating from a marking* $\mathbf{r}$ if it only consumes tokens from places which contain ω tokens, more precisely, for each place p such that ${}^\bullet t[p] > 0$, we have $\mathbf{r}[p] = \omega$. In other words, a generating transition can only decrease the number of tokens in some place by resetting it; notice that consuming a finite number of tokens from a place that contains ω tokens leaves ω tokens is that place. Suppose that $\mathbf{p} \xrightarrow{t} \mathbf{q} \xrightarrow{t} \mathbf{r}$, where t is a generating transition in $\mathbf{p}$, then $\mathbf{r} \geq \mathbf{q}$. Indeed, if some place is reset by t then by immediately firing t again, the number of tokens in such a place does not decrease below zero. By definition, the number of tokens in places that t only consumes from is ω, both before and after firing a generating transition t. Finally, the number of tokens in all other places can only increase after firing t again. By firing t an arbitrary number of times, the places where t only produces tokens to will then contain ω many tokens.

Formally, $\mathcal{N}$ is the same object as $\mathcal{P}$: it consists of the same sets of places, transitions, and resets, but its semantics differs. A marking $\mathbf{m}$ of $\mathcal{N}$ is allowed to have ω tokens in its places, so $\mathbf{m} \in \mathbb{N}_\omega^n$, where n is the number of places. Recall that ω denotes the first infinite cardinal, so $\omega + z = \omega$ for all $z \in \mathbb{Z}$. To define the semantics of $\mathcal{N}$, we need to specify the behaviour of its transitions. Fix a marking $\mathbf{m}$. As is the case in $\mathcal{P}$, a transition t can be fired in $\mathcal{N}$ if, for every place p, $\mathbf{m}[p] \geq {}^\bullet t[p]$. The marking reached depends on whether t is generating from $\mathbf{m}$. If t is not generating from $\mathbf{m}$, then its behaviour is defined as it was in $\mathcal{P}$; first subtract ${}^\bullet t$, then perform the resets, and lastly add $t^\bullet$. Otherwise, if t is generating from $\mathbf{m}$, then $\mathbf{m} \xrightarrow{t} \mathbf{n}$ is defined so that

$$\mathbf{n}[p] = \begin{cases} \omega & \text{if } p \notin t^\circ, \text{ and either } t^\bullet[p] \geq 1 \text{ or } \mathbf{m}[p] = \omega; \\ t^\bullet[p] & \text{if } p \in t^\circ; \\ \mathbf{m}[p] & \text{otherwise.} \end{cases}$$

Intuitively, the transition is applied arbitrarily many times producing ω tokens to some places, whenever it is possible. Claim 3.3 allows us to instead decide the coverability instance in $\mathcal{N}$ with the abstracted space of configurations.

▷ Claim 3.3. Let $\mathbf{m}, \mathbf{n} \in \mathbb{N}^n$. Then $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{P}$ if and only if $\mathbf{n}$ is coverable from $\mathbf{m}$ in $\mathcal{N}$.

Recall the norm of a marking $\|\mathbf{v}\| = \sum_{\mathbf{v}[p] \in \mathbb{N}} |\mathbf{v}[p]|$. Claim 3.4, shows that because $\mathcal{N}$ is acyclic, only markings with an exponential norm can be reached. Note, critically, that places containing ω tokens do not contribute to the norm.

▷ Claim 3.4. Let k be the greatest number of tokens produced by a transition in $\mathcal{P}$ and let $C = \|\mathbf{m}\|$. If $\mathbf{m} \xrightarrow{*} \mathbf{v}$ in $\mathcal{N}$, then $\|\mathbf{v}\| \leq C \cdot k^n$.

Proof of Theorem 3.2. By Claim 3.3, it suffices to show that coverability in the modified acyclic Petri net with resets $\mathcal{N}$ can be decided in polynomial space. We can do this by non-deterministically exploring the markings that are reachable from $\mathbf{m}$ in $\mathcal{N}$. Given Claim 3.4, if $\mathbf{v}$ is reachable from $\mathbf{m}$ in $\mathcal{N}$, then $\|\mathbf{v}\| \leq C \cdot k^n$. These reachable markings can be written down using polynomially many bits since n is the number of places, k is the greatest number of tokens produced by a place, and C is the norm of $\mathbf{m}$. Thus, coverability in acyclic Petri nets with resets is in PSPACE. ◀

4 Lower Bounds

4.1 Coverability in Acyclic Workflow Nets with Resets

With the coverability objective, binary-encoded transitions can be weakly simulated by unary-encoded transitions. We do this for convenience, since the later reductions can be more succinctly presented with binary-encoded transitions.

▶ **Lemma 4.1.** *Given be an instance of coverability in acyclic workflow nets with resets with binary-encoded transitions $I = (\mathcal{B}, \mathbf{m}, \mathbf{n})$, one can construct, in polynomial time, an instance of coverability in acyclic workflow nets with resets $I' = (\mathcal{U}, \mathbf{x}, \mathbf{y})$ with unary-encoded transitions such that I is positive if and only if I' is positive.*

In this section we prove the following theorem. The proofs of all auxiliary claims and lemmata can be found in the full version of the paper.

▶ **Theorem 4.2.** *Coverability in acyclic workflow nets with resets is* PSPACE*-hard.*

Proof Approach

We will reduce from the QSAT problem.

QSAT
INPUT: A QBF φ in CNF over variables $y_1, x_1, \ldots, y_k, x_k$.
QUESTION: Does $\forall y_1 \, \exists x_1 \ldots \forall y_k \, \exists x_k : \varphi(y_1, x_1, \ldots, y_k, x_k)$ evaluate to true?

Given a Quantified Boolean Formula (QBF) φ in Conjunctive Normal Form (CNF), we will construct an acyclic workflow net with resets $\mathcal{W}$ that mimics the exhaustive approach to verifying φ. There will be a collection of places that represent an assignment to the variables $y_1, x_1, \ldots, y_k, x_k$. There will be transitions that consume tokens from these places and produce tokens into a component of $\mathcal{W}$ that is used to test whether the current assignment is satisfying. If the current assignment is satisfying, then one token can be produced to some final place which counts the number of satisfying assigments observed. The places representing an assignment are controlled by a series of gadgets that exhaustively iterate through each possible assignment to the universal variables and allow for nondeterministic assignment to the existential variables. A marking in which the final place contains 2^k tokens can only be reached if and only if every considered assignment has been checked to be satisfying. A detailed description of the coverability instance follows. The proof of correctness consists of two parts.

First, we will verify that the QBF evaluates to true given that coverability holds. We achieve this via an inductive argument that tracks the simulated assignments to variables over parts of the run witnessing coverability.

In the second part, we would like to recover a firing sequence for coverability if the QBF evaluates to true. We achieve this by using (partial) assignments to variables in the QBF to inform which transitions need be fired to make progress towards the final marking.

Construction of the Acyclic Workflow Net with Resets

For this section, we focus our attention on a QBF

$$\forall y_1\, \exists x_1\, \forall y_2\, \exists x_2\, \ldots\, \forall y_k\, \exists x_k : \varphi(y_1, x_1, y_2, x_2, \ldots, y_k, x_k).$$

We remark that we can add "dummy" clauses $(\overline{y}_i \vee y_i)$ and $(\overline{x}_i \vee x_i)$ for each $i \in [1, k]$ to φ without changing any valuation.

For the proof of Theorem 4.2, we construct an acyclic workflow net with resets $\mathcal{W} = (P, T, F, R)$ from the QBF; we first list the places and transitions including resets of $\mathcal{W}$. See Figure 4 for an example.

The places. There is a place for each literal: for every $i \in [1, k]$, there is b_i for y_i, $\overline{b}_i$ for $\overline{y}_i$, a_i for x_i, and $\overline{a}_i$ for $\overline{x}_i$. Let L denote the set of the literal places. The non-emptiness of the place $\overline{b}_i$, for example, will represent assigning false to the variable y_i.

There is a place for each clause: for every $j \in [1, m]$, there is c_j for the j-th clause. Furthermore, for every $i \in [1, k]$, there is d_{y_i} for the dummy clause $(y_i \vee \overline{y}_i)$ and there is d_{x_i} for the dummy clause $(x_i \vee \overline{x}_i)$. All *clause places* $c_1, \ldots, c_m, d_{x_1}, d_{y_1}, \ldots, d_{x_k}, d_{y_k}$ are distinct; the set comprising them is denoted C. The non-emptiness of a clause place $c \in C$ will represent whether the corresponding clause has been satisfied.

For each $i \in [1, k]$, there is a *holding place* h_i and a *waiting place* w_i for each universally quantified variable, as well as a *decision place* v_i for each existentially quantified variable. If the holding place h_i contains a token, one should think that the universally quantified variable y_i and all subsequent variables $x_i, y_{i+1}, x_{i+1}, \ldots, y_k, x_k$ have not yet been assigned. The waiting place w_i will contain a token if the universally quantified variable y_i is currently assigned false. The decision place v_i contains a token after the truth assignment of the prior universally quantified variable y_i has completed, but the existentially quantified variable x_i has not yet received a value. The literal places, holding places, waiting places, decision places, and dummy clause places are grouped into *gadgets*. There are k *universal gadgets* $U_i = \{h_i, w_i, b_i, \overline{b}_i, d_{y_i}\}$ and k *existential gadgets* $E_i = \{v_i, a_i, \overline{a}_i, d_{x_i}\}$.

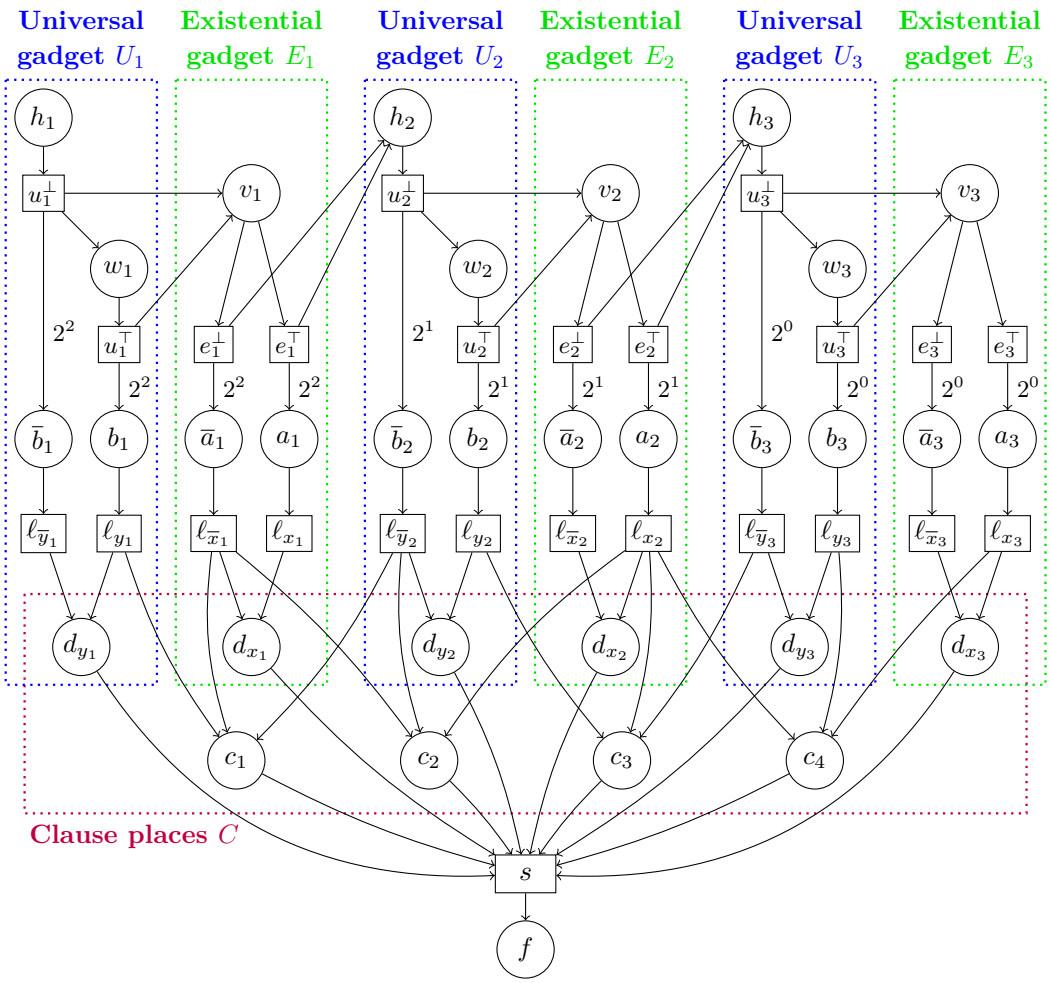


Figure 4 The acyclic workflow net with resets $\mathcal{W}$, drawn without resets for sake of clarity, for the QBF $\forall y_1 \exists x_1 \forall y_2 \exists x_2 \forall y_3 \exists x_3 : \varphi(y_1, x_1, y_2, x_2, y_3, x_3)$ where $\varphi(y_1, x_1, y_2, x_2, y_3, x_3) = (y_1 \vee \overline{x}_1 \vee \overline{y}_2) \wedge (\overline{x}_1 \vee \overline{y}_2 \vee x_2) \wedge (y_2 \vee x_2 \vee \overline{y}_3) \wedge (x_2 \vee y_3 \vee x_3) \wedge (y_1 \vee \overline{y}_1) \wedge (x_1 \vee \overline{x}_1) \wedge (y_2 \vee \overline{y}_2) \wedge (x_2 \vee \overline{x}_2) \wedge (y_3 \vee \overline{y}_3) \wedge (x_3 \vee \overline{x}_3)$. All universal and existential control transitions reset all later occurring places in the universal and existential gadgets and in all clause places. The loading transitions reset all later occurring dummy clause places. The satisfaction transition resets all clause places.

Finally, there is a place f which counts the number of assignments that have been verified to satisfy the QBF.

The initial place i of the workflow is h_1 and the final place f of the workflow is f.

The transitions. Here, binary-encoded transitions are used, see Lemma 4.1. The resets will be specified later.

Inside the universal gadget U_i, there are two *universal control transitions* $u_i^\perp$ and $u_i^\top$. Firing $u_i^\perp$ corresponds to setting y_i to false, and firing $u_i^\top$ corresponds to setting y_i to true. The transition $u_i^\perp$ consumes one token from h_i, produces one token to w_i, produces one token to v_i, and produces 2^{k-i} tokens to $\overline{b}_i$; the transition $u_i^\top$ consumes one token from w_i, produces one token to v_i, and produces 2^{k-i} tokens to b_i.

Inside the existential gadget E_i, there are two *existential control transitions* $e_i^\perp$ and $e_i^\top$. Firing $e_i^\perp$ corresponds to setting x_i to false, and firing $e_i^\top$ corresponds to setting x_i to true. The transition $e_i^\perp$ consumes one token from v_i, produces 2^{k-i} tokens to $\overline{a}_i$, and produces one token to h_{i+1}; similarly, the transition $e_i^\top$ consumes one token from v_i, produces 2^{k-i} tokens to a_i, and produces one token to h_{i+1}.

Informally, the i-th universal or existential controlling transitions produce 2^{k-i} tokens to places $\overline{b}_i$, b_i, $\overline{a}_i$, and a_i so that their values are "remembered" whilst the inner quantified variables have their assignments exhausted.

Connecting the universal and existential gadgets to the clause places are a series of *loading transitions*. There is a loading transition for each literal; for each $i \in [1, k]$, there are transitions $\ell_{\overline{y}_i}$, ℓ_{y_i}, $\ell_{\overline{x}_i}$, and ℓ_{x_i}. The loading transition ℓ_{y_i}, for example, consumes a token from the place b_i and produces a token to each clause place corresponding to a clause containing the literal y_i, including the dummy clause place d_{y_i}.

There is a *satisfaction transition* s that consumes a token from each of the clause places and produces a token into a final place f. Intuitively, s can only be fired when all of the clauses have been satisfied (and f is used to count the number of satisfying assignments).

Ordering places and transitions. The following linear ordering *earlier than* (denoted $\prec$) on $P \cup T$ shows that $\mathcal{W}$ is acyclic:

$$h_1, u_1^\perp, w_1, u_1^\top, \overline{b}_1, b_1, v_1, e_1^\perp, e_1^\top, \overline{a}_1, a_1, \ldots, h_k, u_k^\perp, w_k, u_k^\top, \overline{b}_k, b_k, v_k, e_k^\perp, e_k^\top, \overline{a}_k, a_k,$$
$$\ell_{\overline{y}_1}, \ell_{y_1}, \ell_{\overline{x}_1}, \ell_{x_1}, \ldots, \ell_{\overline{y}_k}, \ell_{y_k}, \ell_{\overline{x}_k}, \ell_{x_k}, d_{y_1}, d_{x_1}, \ldots, d_{y_k}, d_{x_k}, c_1, \ldots, c_m, s, f.$$

The resets. The universal and existential control transitions reset all *later* occurring places in the universal gadgets and existential gadgets and *all* dummy clause places. This also includes the places corresponding to the literals; for example, $u_i^\perp$ and $u_i^\top$ reset both $\overline{b}_i$ and b_i, so it is always true that either $\overline{b}_i$ and b_i is empty.

This effectively forces the universal and existential control transitions to be fired in sequence: $u_1^\perp$ or $u_1^\top$, then $e_1^\perp$ or $e_1^\top$, then $u_2^\perp$ or $u_2^\top$, etc., until $e_k^\perp$ or $e_k^\top$ is fired.

The loading transitions reset all later occurring dummy clause places. For example, ℓ_{y_i} resets $d_{x_i}, d_{y_{i+1}}, d_{x_{i+1}}, \ldots, d_{y_k}, d_{x_k}$. Similarly, this forces the loading transitions to also be fired in sequence: $\ell_{\overline{y}_1}$ or ℓ_{y_1}, then $\ell_{\overline{x}_1}$ or ℓ_{x_1}, then $\ell_{\overline{y}_2}$ or ℓ_{y_2}, until $\ell_{\overline{x}_k}$ or ℓ_{x_k} is fired. This is due to the fact that all dummy places must be non-empty to fire the satisfaction transition.

Finally, the satisfaction transition resets all clause places. It could be the case that a clause contains two true literals under an assignment, so the clause place contains two tokens. It is necessary to clear such a place. Note that the final place f cannot be reset.

Coverability instance $(\mathcal{W}, \mathbf{m}, \mathbf{n})$. We have just defined the acyclic workflow net with resets $\mathcal{W}$. The initial marking $\mathbf{m}$ only has one token in the initial place; $\mathbf{m}[h_1] = 1$ and, for all $p \in P \setminus \{h_1\}, \mathbf{m}[p] = 0$. The target marking $\mathbf{n}$ only has 2^k tokens in the final place; $\mathbf{n}[f] = 2^k$ and, for all $p \in P \setminus \{f\}, \mathbf{n}[p] = 0$.

Part One: Coverability implies QBF is true

We would like to prove an inductive statement of the following, informally described, kind. Consider any run from the initial marking that covers the target marking. Let σ be an infix of this run from $\mathbf{p}$ to $\mathbf{q}$, and let i be a number in $[0, k]$ such that 2^i divides $\mathbf{p}[f]$ and that $\mathbf{q}[f] = \mathbf{p}[f] + 2^i$. This means that σ fires the satisfaction transition, s, 2^i many times. Then the following (partial) QBF is true:

$$\forall y_{k-i+1}\ \exists x_{k-i+1} \ldots \forall y_k\ \exists x_k : \varphi(\beta_1, \alpha_1, \ldots, \beta_{k-i}, \alpha_{k-i}, y_{k-i+1}, x_{k-i+1}, \ldots, y_k, x_k).$$

Here $(\beta_1, \alpha_1, \ldots, \beta_{k-i}, \alpha_{k-i}) \in \{0,1\}^{2(k-i)}$ are determined by $\mathbf{p}$.

To realise this plan, we need several ingredients. For the base case of the induction, $i = 0$: σ only fires s once. We will determine $(\beta_1, \alpha_1, \beta_2, \alpha_2, \ldots, \beta_k, \alpha_k) \in \{0,1\}^{2k}$ based on $\mathbf{p}$, in particular on which of the places $\overline{b}_i$ and b_i, as well as $\overline{a}_i$ and a_i, are non-empty in $\mathbf{p}$. Note that it might not be sufficient to consider only the marking $\mathbf{p}$ since this could be, for instance, the initial marking $\mathbf{m}$, which has all places empty, bar h_1. So the "existential decisions" that determine $\alpha_1, \alpha_2, \ldots, \alpha_k$ need to be found from a prefix of σ.

For the inductive step, $i > 0$: the infix σ fires the satisfaction transition 2^i times. We will split σ in two: σ_0 and σ_1. We will use the inductive hypothesis on both subruns. For this to work, we will show that the partial assignments

$$(\beta_1, \alpha_1, \ldots, \beta_{k-i+1}, \alpha_{k-i+1}) \in \{0,1\}^{2(k-i+1)} \qquad \text{and}$$
$$(\beta'_1, \alpha'_1, \ldots, \beta'_{k-i+1}, \alpha'_{k-i+1}) \in \{0,1\}^{2(k-i+1)},$$

which are determined based on each half of the run, satisfy the constraints

$$\beta_1 = \beta'_1,\ \alpha_1 = \alpha'_1,\ \ldots,\ \beta_{k-i} = \beta'_{k-i},\ \alpha_{k-i} = \alpha'_{k-i}, \quad \beta_{k-i+1} = 0, \quad \text{and} \quad \beta'_{k-i+1} = 1.$$

Informally speaking, these partial assignments are complementary with respect to the i-th innermost universally quantified variable. Note that the index variable i is reused in a variety of contexts throughout the following claims.

Properties of markings.

▷ Claim 4.3. If $\mathbf{v}$ is reachable from $\mathbf{m}$, then for every $i \in [1,k]$, $\mathbf{v}[\overline{b}_i] = 0$ or $\mathbf{v}[b_i] = 0$, and $\mathbf{v}[\overline{a}_i] = 0$ or $\mathbf{v}[a_i] = 0$.

▷ Claim 4.4. If $\mathbf{p} \xrightarrow{t} \mathbf{q}$, then $\mathbf{q}[f] - \mathbf{p}[f] \in \{0,1\}$.

Let us define, for each $i \in [1,k]$, two functions $g_i, g'_i : \mathbb{N}^P \to \mathbb{N}$ that map a marking to a natural number. We will use these functions to define a collection of *good* markings.

$$g_i(\mathbf{v}) \coloneqq \mathbf{v}[f] + \mathbf{v}[\overline{b}_i] + \mathbf{v}[b_i] + \mathbf{v}[d_{y_i}] + \sum_{j=1}^{i} 2^{k-j} \cdot (2\mathbf{v}[h_j] + \mathbf{v}[w_j] + \mathbf{v}[v_j]) - 2^{k-i} \cdot \mathbf{v}[v_i]$$

$$g'_i(\mathbf{v}) \coloneqq \mathbf{v}[f] + \mathbf{v}[\overline{a}_i] + \mathbf{v}[a_i] + \mathbf{v}[d_{x_i}] + \sum_{j=1}^{i} 2^{k-j} \cdot (2\mathbf{v}[h_j] + \mathbf{v}[w_j] + \mathbf{v}[v_j])$$

▶ **Definition 4.5** (Good marking). *A marking* $\mathbf{v}$ *is* good *if for each* $i \in [1,k]$, $g_i(\mathbf{v}) = 2^k$ *and* $g'_i(\mathbf{v}) = 2^k$. *A marking is* bad *if it is not good.*

Roughly speaking, a marking is good if no tokens in the universal gadgets U_i and no tokens in the existential gadgets E_i have been lost due to a reset. We discuss good markings in more detail in the full version of the paper.

▷ Claim 4.6. Suppose $\mathbf{p} \xrightarrow{t} \mathbf{q}$, then $g_i(\mathbf{p}) \geq g_i(\mathbf{q})$ and $g'_i(\mathbf{p}) \geq g'_i(\mathbf{q})$ for each $i \in [1,k]$.

▷ Claim 4.7. Suppose $\mathbf{p} \xrightarrow{t} \mathbf{q}$, where $\mathbf{p}$ is reachable from $\mathbf{m}$. If $\mathbf{q}$ is good, then $\mathbf{p}$ is good.

Given Claim 4.7 and since the target marking $\mathbf{n}$ is good, only good markings can be observed on a covering run from the initial marking $\mathbf{m}$. From this, we know that if a bad marking is ever reached, the target marking cannot be covered.

▷ Claim 4.8. If $\mathbf{m} \xrightarrow{\pi} \mathbf{n}'$ where $\mathbf{n}' \geq \mathbf{n}$, then $\mathbf{n}' = \mathbf{n}$.

The following claim shows that resetting any non-empty place in any of the universal or existential gadgets results in a bad marking. Recall $\prec$, the previously defined *earlier than* ordering of places and transitions.

▷ Claim 4.9. Suppose $\mathbf{p} \xrightarrow{t} \mathbf{q}$ where $\mathbf{p}$ is reachable and $t \in \{u_i^\perp, u_i^\top, e_i^\perp, e_i^\top : i \in [1,k]\}$. If there exists $p \in U_1 \cup E_1 \cup \cdots \cup U_k \cup E_k$ such that $t \prec p$ and $\mathbf{p}[p] \geq 1$, then $\mathbf{q}$ is bad.

Extracting Assignments from Markings

We will now explain the relationship between markings and partial assignments. For a good marking $\mathbf{v}$, let $\mathrm{val}(\mathbf{v})$ be the vector $(\beta_1, \alpha_1, \beta_2, \alpha_2, \ldots, \beta_k, \alpha_k) \in \{0, 1, ?\}^{2k}$ such that

$$\beta_i := \begin{cases} 0 & \mathbf{v}[\overline{b}_i] \geq 1 \\ 1 & \mathbf{v}[b_i] \geq 1 \\ ? & \text{otherwise,} \end{cases} \quad \text{and} \quad \alpha_i := \begin{cases} 0 & \mathbf{v}[\overline{a}_i] \geq 1 \\ 1 & \mathbf{v}[a_i] \geq 1 \\ ? & \text{otherwise.} \end{cases}$$

The intention is that, for every $i \in [1,k]$, β_i and α_i correspond to the values of the Boolean variables y_i and x_i, respectively. Note that Claim 4.3 ensures that β_i and α_i are well-defined, since, for example, $\overline{b}_i$ and b_i cannot both be non-empty in a reachable marking. Notice that not all good markings correspond to fully defined variable assignments, but only those in which all h_i and v_i are empty. We will see that $\mathbf{p}[h_i] = \mathbf{p}[v_i] = 0$ implies that either $\overline{b}_i$ or b_i and either $\overline{a}_i$ or a_i are non-empty, except for right at the end, for example when the target marking $\mathbf{n}$ is reached. To take an example, if h_i contains a token, then neither $\overline{b}_i$ nor b_i will contain a token. Informally, this can be interpreted as thinking that the Boolean variable y_i has not yet been assigned its value; only after firing $u_i^\perp$ does it first get assigned false (before later being assigned true when $u_i^\top$ is eventually fired).

Recall that $C \subseteq P$ is the collection of clause places. We say that a marking $\mathbf{v}$ is *clause-free* if $\mathbf{v}[c] = 0$ for all $c \in C$.

▶ **Lemma 4.10.** *Fix $i \in [0,k]$ and suppose $\mathbf{p} \xrightarrow{\sigma} \mathbf{q}$ and the following properties hold:*

(1) *$\mathbf{p}$ is a clause-free marking that is reachable from $\mathbf{m}$,*
(2) *$\mathbf{n}$ is coverable from $\mathbf{q}$,*
(3) *2^i divides $\mathbf{p}[f]$ and $\mathbf{q}[f] = \mathbf{p}[f] + 2^i$,*
(4) *the last transition of σ is s,*
(5) *for all $j \in [1, k-i]$, $\mathbf{p}[\overline{b}_i] + \mathbf{p}[b_i] \geq 2^i$ and $\mathbf{p}[\overline{a}_i] + \mathbf{p}[a_i] \geq 2^i$,*
(6) *if $i > 0$, then $\mathbf{p}[h_{k-i+1}] = 1$, and*
(7) *if $i > 0$, then, for all $p \in U_{k-i+1} \cup E_{k-i+1} \cup \cdots \cup U_k \cup E_k$ except h_{k-i+1}, $\mathbf{p}[p] = 0$.*

Let $\mathrm{val}(\mathbf{p}) = (\beta_1, \alpha_1, \ldots, \beta_k, \alpha_k)$. Then the following QBF evaluates to true:

$$\forall y_{k-i+1}\, \exists x_{k-i+1} \ldots \forall y_k\, \exists x_k : \varphi(\beta_1, \alpha_1, \ldots, \beta_{k-i}, \alpha_{k-i}, y_{k-i+1}, x_{k-i+1}, \ldots, y_k, x_k).$$

Moreover, σ does not fire transitions $u_1^\perp, u_1^\top, e_1^\perp, e_1^\top, \ldots, u_{k-i}^\perp, u_{k-i}^\top, e_{k-i}^\perp, e_{k-i}^\top$.

Part Two: QBF is true implies Coverability

Here we would like to recover a firing sequence for coverability if the QBF evaluates to true. Depending on the current assignment of the universally quantified variables, $y_1, \dots y_i$, and the already selected assignments of the existentially quantified variables $x_1, \dots, x_{i-1}$, one can use the truth of the QBF to determine whether x_i is assigned true or false, this informs which of the next existentially quantified transitions to fire.

▶ **Lemma 4.11.** *Fix $i \in [0, k]$ and suppose that for some $\beta_1, \alpha_1, \dots, \beta_{k-i}, \alpha_{k-i} \in \{0, 1\}$, the following QBF evaluates to true.*

$$\forall y_{k-i+1}\, \exists x_{k-i+1} \dots \forall y_k\, \exists x_k : \varphi(\beta_1, \alpha_1, \dots, \beta_{k-i}, \alpha_{k-i}, y_{k-i+1}, x_{k-i+1}, \dots, y_k, x_k)$$

Let $\mathbf{p}$ be a marking such that, if $i > 0$ then $\mathbf{p}[h_{k-i+1}] = 1$, and for every $j \in [1, k-i]$,
(1) $\mathbf{p}[\overline{b}_j] \geq 2^i$ *if* $\beta_j = 0$, *otherwise* $\mathbf{p}[b_j] \geq 2^i$ *if* $\beta_j = 1$,
(2) $\mathbf{p}[\overline{a}_j] \geq 2^i$ *if* $\alpha_j = 0$, *otherwise* $\mathbf{p}[a_j] \geq 2^i$ *if* $\alpha_j = 1$.
Then there exists a firing sequence σ such that $\mathbf{p} \xrightarrow{\sigma} \mathbf{q}$ where $\mathbf{q}$ is a marking such that $\mathbf{q}[f] = \mathbf{p}[f] + 2^i$ and for every $j \in [1, k-i]$,
(a) $\mathbf{q}[\overline{b}_j] + \mathbf{q}[b_j] = \mathbf{q}[\overline{b}_j] + \mathbf{q}[b_j] - 2^i$,
(b) $\mathbf{q}[\overline{a}_j] + \mathbf{q}[a_j] = \mathbf{q}[\overline{a}_j] + \mathbf{q}[a_j] - 2^i$, *and*
(c) $\mathbf{q}[h_j] = \mathbf{p}[h_j]$, $\mathbf{q}[w_j] = \mathbf{p}[w_j]$, *and* $\mathbf{q}[v_j] = \mathbf{p}[v_j]$.

Completing the proof

Proof of Theorem 4.2. The reduction from QSAT is already outlined above. Given an instance of QSAT that consists of a QBF φ over $y_1, x_1, \dots, y_k, x_k$, there exists an instance of coverability in acyclic workflow nets with resets $(\mathcal{W}, \mathbf{m}, \mathbf{n})$ such that $\forall y_1\, \exists x_1 \dots \forall y_k\, \exists x_k : \varphi(y_1, x_1, \dots, y_k, x_k)$ evaluates to true if and only if $\mathbf{m} \xrightarrow{*} \mathbf{n}'$ in $\mathcal{W}$ where $\mathbf{n}' \geq \mathbf{n}$. The backwards implication is given by Lemma 4.10 with $i = k$, $\mathbf{p} = \mathbf{m}$, and $\mathbf{q} = \mathbf{n}'$. The forwards implication is given by Lemma 4.11 with $i = k$. ◀

▶ **Corollary 4.12.** *Reachability in acyclic workflow nets with resets and coverability in both acyclic Petri nets with resets and acyclic workflow net with resets are all* PSPACE*-complete.*

4.2 Reachability in Acyclic Petri Nets with Resets

In this section, we will prove that reachability in acyclic Petri nets with resets is undecidable. We reduce from reachability in Petri nets with zero tests, a problem that is well-known to be undecidable [1], following from the undecidability of reachability in counter machines [21]. A *Petri net with zero tests* is a tuple (P, T, F, Z), where (P, T, F) is a Petri net and $Z : T \to 2^P$ is a function defining the zero-test edges. A transition $t \in T$ zero-tests a place $p \in P$ if $p \in Z(t)$. Then t can be fired only if p is empty. As is the case for resets, an *acyclic Petri net with zero tests* does not subject zero-test edges to the acyclicity restriction.

▶ **Theorem 4.13.** *Reachability in acyclic Petri nets with resets is undecidable.*

The reduction is split into two parts. Lemma 4.14 shows how acyclic Petri nets with zero tests can simulate (not necessarily acyclic) Petri nets with zero tests. This requires using zero tests and transitions that do not consume tokens and transitions that do not produce tokens. Then, in Lemma 4.15, we show how acyclic Petri nets with resets can simulate acyclic Petri nets with zero tests. This requires some additional places and relies on the reachability objective to ensure that zero tests are simulated faithfully. The proof is very similar to the proof that reachability in Petri nets with resets is undecidable. We follow through with the construction to make it clear that acyclicity is preserved.

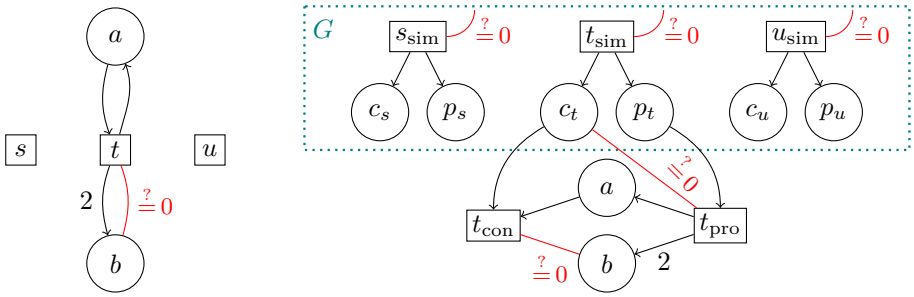


Figure 5 Suppose there is a Petri net with zero tests with transitions s, t, and u. Left: part of the Petri net with zero tests concerning the transition t. The consumptions and productions of s and u are not shown for simplicity. This Petri net is *not* acyclic since t both consumes 1 token from and produces 1 token to a. Right: part of the equivalent acyclic Petri net with zero tests. Places in G are shown, as well as transitions s_{sim}, t_{sim}, and u_{sim} for choosing the next transition to be fired. Since the consumptions and productions of s and u are not shown for simplicity, we also omit the corresponding s_{con}, s_{pro}, u_{con}, and u_{pro}. Importantly, zero-test edges between t_{con} and b, and between t_{pro} and c_t are not subject to the acyclicity restriction. Zero-test edges incident to s_{sim}, t_{sim}, and u_{sim} indicate that all places in G are zero-tested.

▶ **Lemma 4.14.** *The reachability problem in Petri nets with zero tests is reducible in logarithmic space to the reachability problem in acyclic Petri nets with zero tests.*

Proof. Let $\mathcal{P} = (P, T, F, Z)$ be a Petri net with zero tests. We will construct an acyclic Petri net with zero tests $\mathcal{Z} = (P', T', F', Z')$. For every transition $t \in T$, we will add two additional places c_t and p_t to the set of places. Formally, we define $G = \{c_t, p_t \mid t \in T\}$ and $P' = P \cup G$. For every transition $t \in T$, we create three transitions t_{sim}, t_{con}, and t_{pro}, so $T' = \{t_{\text{sim}}, t_{\text{con}}, t_{\text{pro}} \mid t \in T\}$. The intention is that firing $t \in T$ will be simulated by firing t_{sim}, t_{con}, and t_{pro} successively. Figure 5 illustrates the construction. To define the transitions in detail, fix $t \in T$.

- The transition t_{sim} simulates choosing t to be the next transition. Formally, $t_{\text{sim}}^{\bullet}[c_t] = t_{\text{sim}}^{\bullet}[p_t] = 1$, and $Z'(t_{\text{sim}}) = G$. Note that to fire t_{sim} all places in G must be empty, and upon firing t_{sim}, a token is placed in c_t and p_t. Thus no other transition s_{sim}, t_{sim}, or u_{sim} can be fired until the tokens in c_t and p_t are consumed.
- The transition t_{con} performs the token consumption and zero tests of t. Formally, ${}^{\bullet}t_{\text{con}}[c_t] = 1$, $Z'(t_{\text{con}}) = Z(t)$, and ${}^{\bullet}t_{\text{con}}[p] = {}^{\bullet}t[p]$ for every $p \in P$. The consumption of the token from c_t indicates that the consumptions and zero tests of t have been actioned.
- The transition t_{pro} performs the token productions of t. Formally, ${}^{\bullet}t_{\text{pro}}[p_t] = 1$, $Z'(t_{\text{con}}) = \{c_t\}$, and $t_{\text{pro}}^{\bullet}[p] = t^{\bullet}[p]$, for each $p \in P$. The consumption of the token from p_t indicates that the productions of t have been actioned. The zero test on c_t forces a firing order that mimics the semantics of firing t.

Indeed, after firing t_{sim} the only transition that can be fired is t_{con} since all other transitions require c_t to be empty or require a place in $G \setminus \{c_t\}$ to be non-empty. Then, after firing t_{con} the only transition that can be fired is t_{pro} since all other transitions either require p_t to be empty, or require a place in $G \setminus \{p_t\}$ to be non-empty.

Given a marking $\mathbf{v}$ over P, define $\mathbf{v}'$ over P' such that $\mathbf{v}'[p] = \mathbf{v}[p]$ for all $p \in P$ and $\mathbf{v}'[q] = 0$ for all $q \in G$. It follows that $\mathbf{m} \xrightarrow{*} \mathbf{n}$ in $\mathcal{P}$ if and only if $\mathbf{m}' \xrightarrow{*} \mathbf{n}'$ in $\mathcal{Z}$. Indeed, runs in $\mathcal{P}$ have equivalent runs in $\mathcal{Z}$, where each firing of a transition t is replaced with the firing of transitions t_{sim}, then t_{con}, then t_{pro}. Conversely, as previously detailed, runs in $\mathcal{Z}$ must fire t_{sim}, t_{con}, and t_{pro} successively for some transition $t \in T$.

It remains to observe that $\mathcal{Z}$ is acyclic. Consider the following ordering: places in G occur before production transitions (such as t_{pro}), which occur before places in P, which occur before consumption transitions (such as t_{con}). ◀

▶ **Lemma 4.15.** *The reachability problem for acyclic Petri nets with zero tests is reducible in logarithmic space to the reachability problem for acyclic Petri nets with resets.*

Proof of Lemma 4.15. Via leveraging the reachability objective, the idea is to add a copy of each place that will make sure zero tests are simulated faithfully.

Let $\mathcal{Z} = (P, T, F, Z)$ be an acyclic Petri net with zero tests. We will construct an acyclic Petri net with resets $\mathcal{R} = (P', T', F', R)$. For each place $p \in P$ we will add a copy place c_p, so $P' = \{p, c_p : p \in P\}$. For each transition $t \in T$, there will be a corresponding transition $t' \in T'$ with the following behaviour. Firstly, t' will mimic the token consumption and token production between the original places and their copies, so for every place p, ${}^\bullet t'[p] = {}^\bullet t'[c_p] = {}^\bullet t[p]$ and $t'^\bullet[p] = t'^\bullet[c_p] = t^\bullet[p]$. Secondly, suppose a place $p \in P$ is zero-tested by $t \in T$, i.e. $p \in Z(t)$. Then $t' \in T'$ will reset $p \in P'$ but not the copy $c_p \in P'$. Note that none of the copy places are ever reset.

Given the initial marking $\mathbf{m}$ and target marking $\mathbf{n}$ over P, we define $\mathbf{m}'$ and $\mathbf{n}'$ over P' so that $\mathbf{m}[p] = \mathbf{m}'[p] = \mathbf{m}'[c_p]$ and $\mathbf{n}[p] = \mathbf{n}'[p] = \mathbf{n}'[c_p]$. In other words, the markings over P' allocate the same number of tokens to the copy places as their original counterparts. Suppose $\mathbf{m}' \xrightarrow{*} \mathbf{n}'$ in $\mathcal{R}$. Then the invariant $\mathbf{m}'[c_p] - \mathbf{m}'[p] \leq \mathbf{n}'[c_p] - \mathbf{n}'[p]$ holds for all $p \in P$. This inequality is strict only if at some point during the run, a transition is fired that resets a non-empty place. Therefore, $\mathbf{m} \xrightarrow{*} \mathbf{n}$ in $\mathcal{Z}$ if and only if $\mathbf{m}' \xrightarrow{*} \mathbf{n}'$ in $\mathcal{R}$. Indeed, a zero test on p succeeds in $\mathcal{Z}$ if and only if its corresponding reset has no effect, this occurs when p and c_p are empty. To conclude, it is clear that this is a logarithmic-space reduction and that the acyclicity of the consumption and production arcs between places and transitions is preserved in $\mathcal{R}$. ◀

▶ Remark. Lemma 4.14 does not hold with the workflow properties but Lemma 4.15 does; neither holds for the coverability objective.

Proof of Theorem 4.13. Combine Lemma 4.14, Lemma 4.15, and the fact that reachability in Petri nets with zero tests is undecidable. ◀

References

1. Toshiro Araki and Tadao Kasami. Some decision problems related to the reachability problem for Petri nets. *Theor. Comput. Sci.*, 3(1):85–104, 1976. doi:10.1016/0304-3975(76)90067-0.
2. Michael Blondin, Christoph Haase, Filip Mazowiecki, and Mikhail A. Raskin. Affine extensions of integer vector addition systems with states. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/lmcs-17(3:1)2021.
3. Michael Blondin, Filip Mazowiecki, and Philip Offtermatt. The complexity of soundness in workflow nets. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, pages 20:1–20:13. ACM, 2022. doi:10.1145/3531130.3533341.
4. Michael Blondin, Filip Mazowiecki, and Philip Offtermatt. Verifying generalised and structural soundness of workflow nets via relaxations. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification – 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, volume 13372 of *Lecture Notes in Computer Science*, pages 468–489. Springer, 2022. doi:10.1007/978-3-031-13188-2_23.

5 Michael Blondin and Mikhail A. Raskin. The complexity of reachability in affine vector addition systems with states. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/lmcs-17(3:3)2021.

6 Dmitry Chistikov, Christoph Haase, and Simon Halfon. Context-free commutative grammars with integer counters and resets. *Theor. Comput. Sci.*, 735:147–161, 2018. Special issue for RP 2014. doi:10.1016/j.tcs.2016.06.017.

7 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, 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.

8 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.

9 Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer, 1998. doi:10.1007/BFb0055044.

10 Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors, *Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings*, volume 5701 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2009. doi:10.1007/978-3-642-03848-8_19.

11 Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with Dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 269–278. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.39.

12 Alain Finkel, Pierre McKenzie, and Claudine Picaronny. A well-structured framework for analysing Petri net extensions. *Inf. Comput.*, 195(1-2):1–29, 2004. doi:10.1016/j.ic.2004.01.005.

13 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In Mario Bravetti and Gianluigi Zavattaro, editors, *CONCUR 2009 – Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2009. doi:10.1007/978-3-642-04081-8_25.

14 Kunihiko Hiraishi and Atsunobu Ichikawa. A class of Petri nets that a necessary and sufficient condition for reachability is obtainable. *Transactions of the Society of Instrument and Control Engineers*, 24(6):635–640, 1988. doi:10.9746/sicetr1965.24.635.

15 Alexander Kaiser, Daniel Kroening, and Thomas Wahl. A widening approach to multithreaded program verification. *ACM Trans. Program. Lang. Syst.*, 36(4):14:1–14:29, 2014. doi:10.1145/2629608.

16 Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Węgrzycki. Coverability in VASS Revisited: Improving Rackoff's Bound to Obtain Conditional Optimality. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 131:1–131:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2023.131.

17 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

18 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785796.

19 Duan Li, Xiaoling Sun, Jianjun Gao, Shenshen Gu, and Xiaojin Zheng. Reachability determination in acyclic Petri nets by cell enumeration approach. *Autom.*, 47(9):2094–2098, 2011. doi:10.1016/j.automatica.2011.06.017.

20 R. J. Lipton. *The reachability problem requires exponential space.* Research Report. Department of Computer Science, Yale University, 1976. URL: http://www.cs.yale.edu/publications/techreports/tr63.pdf.

21 Marvin L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., 1967.

22 C. A. Petri. Introduction to general net theory. In Wilfried Brauer, editor, *Net Theory and Applications, Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, Germany, October 8-19, 1979*, volume 84 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 1979. doi:10.1007/3-540-10001-6_21.

23 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6:223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

24 Philippe Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In Petr Hlinený and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 616–628. Springer, 2010. doi:10.1007/978-3-642-15155-2_54.

25 Ferucio Laurentiu Tiplea, Corina Bocaneala, and Raluca Chirosca. On the complexity of deciding soundness of acyclic workflow nets. *IEEE Trans. Syst. Man Cybern. Syst.*, 45(9):1292–1298, 2015. doi:10.1109/TSMC.2015.2394735.

26 Ferucio Laurentiu Tiplea and Dan C. Marinescu. Structural soundness of workflow nets is decidable. *Inf. Process. Lett.*, 96(2):54–58, 2005. doi:10.1016/j.ipl.2005.06.002.

27 Rüdiger Valk. Self-modifying nets, a natural extension of Petri nets. In Giorgio Ausiello and Corrado Böhm, editors, *Automata, Languages and Programming, Fifth Colloquium, Udine, Italy, July 17-21, 1978, Proceedings*, volume 62 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 1978. doi:10.1007/3-540-08860-1_35.

28 Wil M. P. van der Aalst. The application of Petri nets to workflow management. *J. Circuits Syst. Comput.*, 8(1):21–66, 1998. doi:10.1142/S0218126698000043.

29 Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets with reset arcs. *Trans. Petri Nets Other Model. Concurr.*, 3:50–70, 2009. doi:10.1007/978-3-642-04856-2_3.

30 Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects Comput.*, 23(3):333–363, 2011. doi:10.1007/s00165-010-0161-4.

Robust Positivity Problems for Linear Recurrence Sequences

The Frontiers of Decidability for Explicitly Given Neighbourhoods

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

Abstract

Linear Recurrence Sequences (LRS) are a fundamental mathematical primitive for a plethora of applications such as the verification of probabilistic systems, model checking, computational biology, and economics. Positivity (are all terms of the given LRS non-negative?) and Ultimate Positivity (are all but finitely many terms of the given LRS non-negative?) are important open number-theoretic decision problems. Recently, the robust versions of these problems, that ask whether the LRS is (Ultimately) Positive despite small perturbations to its initialisation, have gained attention as a means to model the imprecision that arises in practical settings. However, the state of the art is ill-equipped to reason about imprecision when its extent is explicitly specified. In this paper, we consider Robust Positivity and Ultimate Positivity problems where the neighbourhood of the initialisation, expressed in a natural and general format, is also part of the input. We contribute by proving sharp decidability results: decision procedures at orders our techniques are unable to handle for general LRS would entail significant number-theoretic breakthroughs.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Dynamical Systems, Verification, Robustness, Linear Recurrence Sequences, Positivity, Ultimate Positivity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.17

Funding The author is partially funded by DFG grant 389792660 as part of TRR 248 (see `https://perspicuous-computing.science`).

Acknowledgements This work is an extension of a recent collaboration with S Akshay, Hugo Bazille, and Blaise Genest. I am especially grateful to Valérie Berthé for her help in number-theoretic aspects of the paper. Our discussions took place during the Bellairs Dynamical Systems workshop organised by my advisor, Joël Ouaknine. I thank Bellairs Research Institute of McGill University, Barbados, for the hospitality and peaceful environment conducive to scientific discussion. Finally, I thank all the anonymous reviewers for their astute observations and constructive suggestions.

1 Introduction

A real Linear Recurrence Sequence (LRS) of order κ is an infinite sequence of real numbers $(u_0, u_1, u_2, \dots)$ having the following property: there exist κ real constants $a_0, \dots, a_{\kappa-1}$, with $a_0 \neq 0$ such that for all $n \geq 0$:

$$u_{n+\kappa} = a_{\kappa-1}u_{n+\kappa-1} + \cdots + a_0 u_n. \tag{1}$$

The constants $a_0, \dots, a_{\kappa-1}$ define the linear recurrence relation $\mathbf{a}$; they are also associated with the characteristic polynomial $X^\kappa - a_{\kappa-1}X^{\kappa-1} - \cdots - a_1 X - a_0$. The initial terms $u_0, \dots, u_{\kappa-1}$ are collectively denoted as the initialisation $\mathbf{c}$. An LRS is uniquely specified by $(\mathbf{a}, \mathbf{c})$. The best-known example is the Fibonacci sequence $(0, 1, 1, 2, 3, 5, 8, \dots)$, satisfying the recurrence relation $u_{n+2} = u_{n+1} + u_n$: it is named after Leonardo of Pisa, who used it to model the population growth of rabbits. LRS have been extensively studied, and found several mathematical and scientific applications since. The monograph of Everest et al. [15] is a comprehensive treatise on the mathematical aspects of Recurrence Sequences.

© Mihir Vahanwala;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 17; pp. 17:1–17:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Important number-theoretic decision problems for Linear Recurrence Sequences include Positivity (is $u_n \geq 0$ for all n?), Ultimate Positivity (is $u_n \geq 0$ for all but finitely many n?) and the closely related Skolem Problem (is $u_n = 0$ for some n?). We remark that a Positive LRS is necessarily Ultimately Positive. As detailed in [24], these problems have applications in software verification, probabilistic model checking, discrete dynamic systems, theoretical biology, and economics. Decidability has been open for decades, with breakthroughs in restricted settings: Mignotte et al. [21] and Vereshchagin [28] independently proved the Skolem Problem to be decidable up to order 4. Further progress has been reliant on spectral restrictions or number-theoretic conjectures, consult, e.g. [19, 1]. Ouaknine and Worrell [24] showed Positivity and Ultimate Positivity are decidable up to order 5 but number-theoretically hard at order 6. For *simple* LRS (those whose characteristic polynomials have no repeated roots), they showed that Positivity is decidable up to order 9 [23] and Ultimate Positivity is decidable at all orders [25]. These results were originally proven for LRS specified by *rational* recurrences and initialisations, but can be generalised to real algebraic input as well. In this paper, we focus on Positivity and Ultimate Positivity for **sequences defined by real algebraic input**.

In contrast, the *uninitialised* variants of these problems are far more tractable. Braverman [9] and Tiwari [27] consider whether *every* possible initialisation keeps the sequence Positive, and decide so in PTIME. More recently, this result has been extended to processes with choices [4]. We argue that practical applications need a middle ground: recurrence relations that arise in practice need to be contextualised by actual instances of sequences; however, considering *precise* initialisations does not account for inherently imprecise real world measurements, and the requirement of safety margins. We thus study robust variants: given a recurrence and an initialisation, do all initialisations in a neighbourhood satisfy (Ultimate) Positivity?

Related Work

In this paper, we focus on the neighbourhood-of-initialisation notion of robustness, which was first introduced in [2], and more comprehensively treated in [3]. Works with a more control-theoretic flavour include [5], which allows for rounding at every step before applying the recurrence; in the same vein, [12, 13] allow for ε-disturbances at every step of the sequence. Our notion of robustness has been considered in [2, 3, 13], however, these works primarily concern themselves with simply deciding whether there *exists* a neighbourhood around the given point that satisfies Positivity, or whether there *exists* a tolerance ε such that the sequence avoids a region despite ε-disturbances at every step. Although they do identify that robust problems are hard when the neighbourhood is given as input, in the absence of decidability results, their hardness results are not sharp.

There are, of course, broader approaches to model and reason about imprecision: [22] considers a model of computation that can take arbitrary real numbers as input, thereby allowing imprecision in both the initialisation and the recurrence. Even in this setting, the focus is on whether the decision is locally constant in *some* neighbourhood of the given instance of the Positivity Problem, as opposed to whether the decision holds for an entire *given* neighbourhood.

Our contribution

We address the gap in the robustness state of the art by exploring the frontiers of decidability when the neighbourhood is given as input. Concretely, our input consists of a linear recurrence relation $\mathbf{a}$ and a neighbourhood of initialisations centred around $\mathbf{c}$. Our problem is to decide whether all initialisations in the **given** neighbourhood result in (Ultimately) Positive sequences.

When neighbourhoods are expressly given as input, their geometry plays a critical role in the decision procedure. The notion of neighbourhoods that we primarily focus on is based on the ℓ^2-norm. We seek to slightly generalise the study of Euclidean ε-ball neighbourhoods undertaken in [2, 3]. More specifically, we use the Mahalanobis distance to define neighbourhoods. Our parameter is the positive definite matrix $\mathbf{S}$, and the neighbourhood of $\mathbf{c}$ it specifies is the set of all points $\mathbf{c}' \in \mathbb{R}^\kappa$ such that $(\mathbf{c}' - \mathbf{c})^T \mathbf{S}(\mathbf{c}' - \mathbf{c}) \leq 1$. The size of neighbourhoods is usually parametrised by an ε: in our case, we can account for it by simply scaling $\mathbf{S}$. In the statistical context, $\mathbf{S}$ is the inverse of a covariance matrix; and thus, our formulation is a rather natural way of capturing noise and measurement errors in the input, whose components may often be correlated. Our **novelty**, to the best of our knowledge, lies in identifying a general and practical way of explicitly specifying neighbourhoods, and establishing the **first decidability results** in such a setting, albeit at low orders or subject to spectral constraints.

As first discussed in [24, Section 5], solving decision problems on Linear Recurrence Sequences in full generality is an endeavour fraught with number-theoretic hardness. Decision procedures for Positivity problems for LRS of higher order would allow number theorists to compute properties of irrational numbers that are considered inaccessible to contemporary techniques. These include the Diophantine approximation type, which intuitively describes the quality of the "best" rational approximation of a given irrational number, and the Lagrange constant, which intuitively describes how well increasingly precise rational approximations of a given irrational number converge. We justify the inability of our techniques to handle LRS of higher orders by **reducing the computation of Diophantine approximation types and Lagrange constants** to robust Positivity problems for LRS of lower orders than ever before. Prior reductions result in LRS of order 6, we demonstrate that hard instances in our setting exist at order 5. Robust problems for Euclidean ε-ball neighbourhoods were also proven hard at order 6 [2, 3]. The setting we consider generalises this specific case, but we concede that the techniques do not address the decidability status for Euclidean ε-ball neighbourhoods at order 5.

Table 1 Main results, summarised. The distinction between uniform and non-uniform refers to whether the threshold index for certifying Ultimate Positivity must be common for the entire neighbourhood.

	Decidability Proof		
Problem: S-Robust	**General**	**Simple**	**Hardness**
Positivity	order ≤ 4	order ≤ 5	Diophantine hard at order 5
Uniform Ultimate Positivity	order ≤ 4	**all orders**	Lagrange hard at order 5
Non-uniform Ultimate Positivity	order ≤ 4	order ≤ 4	[24, 3]: Lagrange hard at order 6

Structure of the paper

The exponential polynomial closed form is an invaluable tool in the study of LRS, and we devote §2 to its exposition. This equips us to introduce our Robust Positivity Problems and intuit their decidability proofs in §3. Linear Recurrences and Diophantine Approximation are intrinsically connected: number-theoretic results form the basis of decision procedures; open problems are a yardstick for hardness reductions. We survey the number theory relevant to us in §4. We then prove our decidability results in the technical §5 and §6, and present our hardness reduction in §7. We provide concluding perspective in §8. We refer the reader to Appendix A for a summary of the standard notation and prerequisites we use.

2 The exponential polynomial closed form

We begin by discussing the exponential polynomial closed form, a perspective that is routinely leveraged to study the behaviour of Linear Recurrence Sequences. Simple LRS (no repeated characteristic roots) have a surjective correspondence with the closed form

$$u_n = \sum_j w_j \rho_j^n + \sum_j (z_j \gamma_j^n + \bar{z_j} \bar{\gamma_j}^n) \tag{2}$$

where each $\rho_j \in \mathbb{R}, \gamma_j, \bar{\gamma_j} \in \mathbb{C} \backslash \mathbb{R}$ are distinct roots of the characteristic polynomial. By straightforward arithmetic on the above expression, we can see that if $(u_n)_{n \in \mathbb{N}}, (v_n)_{n \in \mathbb{N}}$ are simple LRS with sets of characteristic roots U and V respectively, then

- $r_n = u_n + v_n$ is a simple LRS, whose set of roots is $U \cup V$.
- $r_n = u_n \cdot v_n$ is a simple LRS, whose set of roots is $\{\gamma_1 \gamma_2 : \gamma_1 \in U, \gamma_2 \in V\}$.

In general, one can encode a linear recurrence $\mathbf{a}$ as a $\kappa \times \kappa$ companion matrix $\mathbf{A}$, and interpret the initialisation $\mathbf{c}$ as a vector. Then, u_n is given by the first coordinate of $\mathbf{A}^n \mathbf{c}$, i.e.

$$\begin{bmatrix} u_n \\ u_{n+1} \\ \vdots \\ u_{n+\kappa-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_0 & a_1 & a_2 & \dots & a_{\kappa-1} \end{bmatrix}^n \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{\kappa-1} \end{bmatrix}. \tag{3}$$

Let $\mathbf{e_1}^T$ denote the row vector $\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$. We can thus write $u_n = \mathbf{e_1}^T \mathbf{A}^n \mathbf{c}$. It is now a standard fact that LRS have a surjective correspondence with the following **real exponential polynomial** closed form (every sequence given by the expression is an LRS)

$$u_n = \left(\sum_{j=1}^{k_1} \sum_{\ell=0}^{m_j - 1} z_{j\ell} \rho_j^n n^\ell \right) + \left(\sum_{j=k_1+1}^{k_2} \sum_{\ell=0}^{m_j - 1} (x_{j\ell} \cos n\theta_j + y_{j\ell} \sin n\theta_j) \rho_j^n n^\ell \right) \tag{4}$$

where ρ_j (alternately, $\rho_j e^{i\theta_j}$) are roots of the characteristic polynomial defined by $\mathbf{a}$, each with multiplicity m_j. There are k_1 distinct real roots, and k_2 distinct pairs of complex conjugates among the roots. Given the recurrence relation $\mathbf{a}$, the coefficients $z_{j\ell}, x_{j\ell}, y_{j\ell}$ are each linear functions of $\mathbf{c}$. Indeed, the exponential polynomial solution expresses the sequence as a linear combination of closed-form functions that satisfy the recurrence. For a choice of basis functions, let $\mathbf{q_n}$ denote the vector whose entries are these functions, evaluated at n. In the above example, the entries are of the form $\rho_j n^\ell, \rho_j n^\ell \cos n\theta_j, \rho_j n^\ell \sin n\theta_j$, and so on. The choice of $\{\mathbf{q_n}\}_{n \in \mathbb{N}}$ can differ in "phase": one can replace $\cos n\theta, \sin n\theta$ by $\cos(n\theta - \varphi), \sin(n\theta - \varphi)$ for some choice of φ. Such a choice of basis defines a generalised $\kappa \times \kappa$ Vandermonde matrix $\mathbf{V}$ [16], whose i^{th} row is $\mathbf{q_{i-1}}^T$. By construction, $\mathbf{e_1}^T \mathbf{A}^n \mathbf{V} = \mathbf{q_n}^T$. By defining $\mathbf{p} = \mathbf{V}^{-1} \mathbf{c}$, u_n can thus be equivalently expressed as the inner (dot) product $\langle \mathbf{p}, \mathbf{q_n} \rangle$. The coefficients $z_{j\ell}, x_{j\ell}, y_{j\ell}$ are entries of this very $\mathbf{p}$.

Roots such that $|\rho_j|$ is the largest are called **dominant**. The growth rate of a term in the above expression is governed by $\rho_j^n n^\ell$. Terms with the fastest growth are called **dominant terms**, and they drive the asymptotic behaviour of the LRS. A standard, intuitive prerequisite for Ultimate Positivity is that the leading terms in the exponential polynomial expression must include one that is real and strictly positive, otherwise their dominant contribution oscillates between positive and negative. It is formalised by applying [9, Lemma 4] to the dominant terms in expression 4 and arguing that the contribution from the remaining terms vanishes asymptotically.

▶ **Proposition 1.** *If the characteristic polynomial has no real dominant root of maximum multiplicity, then in any full-dimensional neighbourhood of initialisations, there exists an initialisation, such that the sequence has infinitely many positive terms, and infinitely many negative terms.*

Henceforth, we assume that the characteristic polynomial has a real positive dominant root of maximum multiplicity, for otherwise the answer to Ultimate Positivity is trivially NO. In this paper, we shall always work with algebraic numbers[1], and thus factorising the characteristic polynomial and checking this condition is a standard procedure.

We define $\langle \mathbf{p}, \mathbf{q_n} \rangle_{dom}$ to be the normalised contribution of the dominant terms in the exponential polynomial solution. That is, if the dominant growth rate is $\rho^n n^\ell$, we pick terms with that growth rate, and divide their contribution by $\rho^n n^\ell$. For example, if $u_n = p_1 2^n + p_2 2^n \cos(n\theta - \varphi) + p_3 2^n \sin(n\theta - \varphi) + p_4$ then

$$\langle \mathbf{p}, \mathbf{q_n} \rangle_{dom} = p_1 + p_2 \cos(n\theta - \varphi) + p_3 \sin(n\theta - \varphi).$$

We define $\mu(\mathbf{c}) = \liminf_{n \in \mathbb{N}} \langle \mathbf{p}, \mathbf{q_n} \rangle_{dom}$. Note that μ is an intrinsic property of the initialisation $\mathbf{c}$ and the sequence it generates, and hence is invariant under the choice of "phase shift" φ while defining $\{\mathbf{q_n}\}_{n \in \mathbb{N}}$. In our example, assuming θ is not a rational multiple of π, it is $p_1 - \sqrt{p_2^2 + p_3^2}$

3 Robust Positivity Problems

In this paper, we shall focus on defining and tackling Robust Positivity problems. Our input consists of a linear recurrence relation $\mathbf{a}$, an initialisation $\mathbf{c}$, and a positive definite matrix $\mathbf{S}$ that is used to define a neighbourhood around $\mathbf{c}$. **All input is real algebraic.**

▶ **Problem 1** (**S**-Robust Positivity)**.** *Decide whether for all* $\mathbf{c}'$ *such that* $(\mathbf{c}' - \mathbf{c})^T \mathbf{S} (\mathbf{c}' - \mathbf{c}) \leq 1$*, the LRS* $(\mathbf{a}, \mathbf{c}')$ *is positive.*

▶ **Problem 2** (**S**-Robust Uniform Ultimate Positivity)**.** *Decide whether there exists an* N *such that for all* $\mathbf{c}'$ *with* $(\mathbf{c}' - \mathbf{c})^T \mathbf{S} (\mathbf{c}' - \mathbf{c}) \leq 1$*, the LRS* $(\mathbf{a}, \mathbf{c}')$ *is positive from the* N^{th} *term onwards.*

We can switch the order in which N and $\mathbf{c}'$ are quantified, and query a weaker notion of Robust Ultimate Positivity:

▶ **Problem 3** (**S**-Robust Non-uniform Ultimate Positivity)**.** *Decide whether for all* $\mathbf{c}'$ *with* $(\mathbf{c}' - \mathbf{c})^T \mathbf{S} (\mathbf{c}' - \mathbf{c}) \leq 1$ *, there exists an* N *such that the LRS* $(\mathbf{a}, \mathbf{c}')$ *is positive from the* N^{th} *term onwards.*

The attentive reader might have already noticed that we depart from convention and specify neighbourhoods as *closed* balls. Although [3] does not solve the problems we consider in this paper, it makes crucial observations about the geometry: for Problems 1 and 2, there is no difference between open and closed balls. On the other hand, Problem 3 becomes considerably easier with open balls, and its decidability in this case is tackled in [3] itself.

[1] The field of algebraic numbers $\overline{\mathbb{Q}}$ is the algebraic closure of the rationals $\mathbb{Q}$. Arithmetic and polynomial factoring over $\overline{\mathbb{Q}}$ can be performed with exact precision. We use $\mathbb{A}$ to denote the field of real algebraic numbers, and refer the reader to Appendix A for an initiation to these number fields.

3.1 Uniform Variants: The foundation

In general, an arbitrary point $\mathbf{c}'$ is expressed as $\mathbf{c}+\mathbf{d}$, where $\mathbf{d} \in \mathcal{P}$, a full-dimensional neighbourhood symmetric about the origin. Observe equation 3. The nth term of the LRS is non-negative throughout the neighbourhood if and only if for all $d \in \mathcal{P}$, $\mathbf{e_1}^T\mathbf{A}^n(\mathbf{c}+\mathbf{d}) \geq 0$. We can use the symmetry of $\mathcal{P}$ about the origin to rewrite the above as

$$\mathbf{e_1}^T\mathbf{A}^n\mathbf{c} \geq \max_{\mathbf{d}\in\mathcal{P}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{d} \geq 0. \tag{5}$$

As a simple illustration, assume that the neighbourhood is defined by a polytope rather than a positive definite matrix. This situation arises, for instance, when the metric is based on the ℓ^1- or ℓ^∞-norm, as opposed to the ℓ^2-norm. In this simple example, $\mathcal{P}$ is a polytope, hence $\mathbf{e_1}^T\mathbf{A}^n\mathbf{d}$ is maximised at one of the finitely many corners $\{\mathbf{d_1},\ldots,\mathbf{d_k}\}$. Thus, Robust (Uniform Ultimate) Positivity is decided by using the state of the art [24] to check the (Ultimate) Positivity of each of the LRS $(\mathbf{a}, \mathbf{c}+\mathbf{d_i})$. The geometry of our setting is not simple enough to allow such a straightforward approach. **The overview of our approach to Problems 1 and 2 is as follows.**

1. Decide (constructively for Problem 1) whether there exists an N_1 such that $\mathbf{e_1}^T\mathbf{A}^n\mathbf{c} \geq 0$ for all $n > N_1$. If N_1 is explicitly required, the state of the art is able to tackle LRS of order ≤ 5 [24] and simple LRS satisfying spectral conditions that always hold at orders up to 9 [23]. In the non-constructive case, it can further handle all simple LRS [25].
2. Use linear-algebraic arguments to define a real algebraic LRS $(v_n)_{n=0}^\infty$, such that $v_n \geq 0$ if and only if $|\mathbf{e_1}^T\mathbf{A}^n\mathbf{c}| \geq \max_{\mathbf{d}\in\mathcal{P}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{d}$.
3. Decide (constructively for Problem 1) whether there exists N_2 such that $v_n \geq 0$ for all $n > N_2$. Positivity throughout the neighbourhood is thus guaranteed beyond step $N = \max(N_1, N_2)$. If either N_1 or N_2 does not exist, then Robust Ultimate Positivity, and hence Robust Positivity, does not hold.
4. **Only for Problem 1:** Explicitly check inequality 5 for $n \leq N$.

Our novelty lies in Step 2 and identifying when Step 3 can be implemented. We now discuss how we perform Steps 2 and 4 when $\mathcal{P} = \mathcal{B}_\mathbf{S}$, a neighbourhood of vectors $\mathbf{d}$ such that $\mathbf{d}^T\mathbf{S}\mathbf{d} \leq 1$. The defining parameter $\mathbf{S}$ is a real algebraic positive definite matrix. We note that since $\mathbf{S}$ is positive definite, it can be factored as $\mathbf{G}^T\mathbf{G}$, where $\mathbf{G}$ is a real algebraic invertible matrix. We denote $\mathbf{G}\mathbf{d} = \mathbf{f}$. We argue that $\mathbf{G}^{-1}$ bijectively maps the Euclidean unit ball $\mathcal{B}$ to $\mathcal{B}_\mathbf{S}$. The bijection is clear from the invertibility of the matrix. Suppose $\mathbf{d} = \mathbf{G}^{-1}\mathbf{f}$, where $\mathbf{f} \in \mathcal{B}$, i.e. $\mathbf{f}^T\mathbf{f} \leq 1$. Then $\mathbf{d}^T\mathbf{S}\mathbf{d} = \mathbf{d}^T\mathbf{G}^T\mathbf{G}\mathbf{d} = \mathbf{f}^T\mathbf{f} \leq 1$. Hence,

$$\max_{\mathbf{d}\in\mathcal{B}_\mathbf{S}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{d} = \max_{\mathbf{f}\in\mathcal{B}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{G}^{-1}\mathbf{f}. \tag{6}$$

$\mathcal{B}$ is a convex set; thus a linear function will necessarily be maximised at its boundary, i.e. when $||\mathbf{f}|| = 1$. Given $\mathbf{h}$, the linear function mapping $\mathbf{f}$ to $\mathbf{h}^T\mathbf{f}$ is maximised over the unit Euclidean ball when $\mathbf{f}$ is aligned along $\mathbf{h}$; the maximum value is $||\mathbf{h}||$. We can thus perform Step 4 because

$$\max_{\mathbf{d}\in\mathcal{B}_\mathbf{S}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{d} = \left|\left|\left(\mathbf{e_1}^T\mathbf{A}^n\mathbf{G}^{-1}\right)^T\right|\right|. \tag{7}$$

For Step 2, we need a necessary and sufficient condition for $|\mathbf{e_1}^T\mathbf{A}^n\mathbf{c}| \geq \max_{\mathbf{d}\in\mathcal{B}_\mathbf{S}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{d}$, in terms of the positivity of an LRS at iterate n. We consider the first part of inequality 5, substitute equation 7 in, square both sides, and transfer all terms to the left:

$$(\mathbf{e_1}^T\mathbf{A}^n\mathbf{c})^2 - (\mathbf{e_1}^T\mathbf{A}^n\mathbf{g_1})^2 - \cdots - (\mathbf{e_1}^T\mathbf{A}^n\mathbf{g}_\kappa)^2 \geq 0. \tag{8}$$

Crucially, $\mathbf{g_1}, \ldots, \mathbf{g}_\kappa$ are the linearly independent columns of the invertible $\mathbf{G}^{-1}$. Only Step 3 remains to be addressed: we must (constructively) decide whether there exists N_2 such that the previous inequality holds for all $n > N_2$. In §5, we give the technical details, thus proving our first main decidability result.

▶ **Theorem 2** (First Main Decidability Result). *Problem 2 is decidable for simple LRS. Problem 1 is decidable for simple LRS up to order 5. Problems 1 and 2 are decidable for general LRS up to order 4.*

3.2 The non-uniform variant: An overview

As discussed at length in [2, 3], $\mu(\mathbf{c}') = \liminf_{n \in \mathbb{N}} \langle \mathbf{p}', \mathbf{q_n} \rangle_{dom} \geq 0$ is necessary for the Ultimate Positivity of $\mathbf{c}'$; $\mu(\mathbf{c}') > 0$ is sufficient.

Our strategy for Problem 3 is as follows.

1. Use the First Order Theory of the Reals to check that $\mu(\mathbf{c}') \geq 0$ for all $\mathbf{c}'$ in the given neighbourhood, and detect the critical boundary cases when $\mu(\mathbf{c}') = 0$.
2. Exploit the low dimensionality to decide the critical boundary cases when $\mu(\mathbf{c}') = 0$.

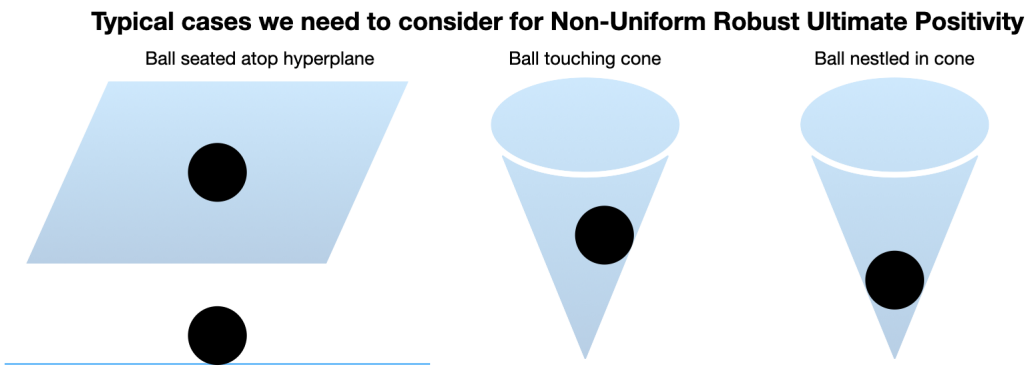


Figure 1 Visual intuition. The region $\mu \geq 0$ is defined by the intersection of halfspaces. The orientation of the neighbourhood relative to this region is deduced with the First Order Theory of the Reals. When there are finitely many halfspaces, the critical case is marked by the ball being tangent to the separating hyperplane(s) at finitely many discrete points. In low dimensions, Ultimate Positivity can be decided for these boundary cases using existing techniques. When there are infinitely many halfspaces, they carve out a region that resembles a cone. The neighbourhood can either touch the cone as before, or be nestled in it, having a continuous, connected region of tangency. In the latter case, Robust Ultimate Positivity can be handled with number-theoretic arguments in the low-dimensional setting.

We adopt this strategy (see Figure 1) and prove our second decidability result in §6.

▶ **Theorem 3** (Second Decidability Result). *Problem 3 is decidable up to order 4.*

4 Diophantine Approximation

We justify the inability of our techniques to generalise to LRS of higher order by establishing a connection to a number-theoretic hurdle: that of Diophantine Approximation. Diophantine Approximation is a vast and active number-theoretic field of research, one of whose concerns is the approximation of reals by rational numbers. A key tool in this regard is the continued fraction expansion $[a_0; a_1, a_2, \ldots]$ of an irrational t:

$$t = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots}}}$$

where $a_0, a_1, a_2, \cdots \in \mathbb{N}$. Truncating this expansion at progressively greater depths yields a series of increasingly accurate approximations. The quality of the rational approximation depends not only on its accuracy but also on the size of the denominator. As discussed in the Introduction, evaluating the quality of the approximation, or that of the convergence, seems inaccessible to contemporary number theory.

The above intuition about the quality of the approximation is captured in the following definition of $L(t)$, the (homogenous) Diophantine approximation type:

$$L(t) = \inf\left\{c \in \mathbb{R} : \left|t - \frac{p}{q}\right| < \frac{c}{q^2} \text{ for some } p, q \in \mathbb{Z}\right\}. \tag{9}$$

Similarly, the quality of the convergence is formalised by defining $L_\infty(t)$, the (homogenous) Lagrange constant:

$$L_\infty(t) = \inf\left\{c \in \mathbb{R} : \left|t - \frac{p}{q}\right| < \frac{c}{q^2} \text{ for infinitely many } p, q \in \mathbb{Z}\right\}. \tag{10}$$

For technical purposes, we use an equivalent definition that relates to the continued fraction perspective, and allows for a slight generalisation. We follow Lagarias and Shallit's terminology [18] and use $[x]$ to denote the shortest distance from x to an integer; for $b \in \mathbb{R}$, $[x]_b$ denotes the shortest distance from x to an integer multiple of b. It is easy to observe the property $[x]_b = b[x/b]$.

▶ **Definition 4** (Diophantine Approximation Type). *The homogenous Diophantine approximation type $L(t)$ is defined to be $\inf_{n\in\mathbb{N}_{>0}} n[nt]$. The inhomogeneous Diophantine approximation type $L(t, s)$ is defined to be $\inf_{n\in\mathbb{N}_{>0}} n[nt - s]$, $s \notin \mathbb{Z} + t\mathbb{Z}$.*

▶ **Definition 5** (Lagrange constant). *The homogenous Lagrange constant $L_\infty(t)$ is defined to be $\liminf_{n\in\mathbb{N}} n[nt]$. The inhomogeneous Lagrange constant $L_\infty(t, s)$ is defined to be $\liminf_{n\in\mathbb{N}} n[nt - s]$, $s \notin \mathbb{Z} + t\mathbb{Z}$.*

From the definitions, it is clear that $0 \le L(t) \le L_\infty(t)$, and also $0 \le L(t, s) \le L_\infty(t, s)$. Due to the work of Khintchine [17], it is known that these constants lie between 0 and $1/\sqrt{5}$. In our setting, the irrational t comes from the argument θ of the characteristic root $\rho e^{i\theta}$ of the LRS: $t = \theta/2\pi$. The following observation that arises from the above fact and properties of the cosine function is pivotal to our novel low-dimensional decidability result for Uniform Robustness.

▶ **Lemma 6.** *For all θ that are not rational multiples of 2π, and all φ, there exist infinitely many $n \in \mathbb{N}$ such that $1 - \cos(n\theta - \varphi) \le \frac{1}{2}\,[n\theta - \varphi]_{2\pi}^2 = 2\pi^2[nt - s]^2 \le \frac{2\pi^2}{5n^2}$.*

The properties of the LRS are driven by whether the characteristic root $\rho e^{i\theta}$ is a scaled root of unity, i.e. θ is a rational multiple of 2π. If yes, decision procedures are often much simpler; if not, we appeal to the number theory discussed in this section. One can detect whether an algebraic characteristic root is a root of unity by brute enumeration.

▶ **Lemma 7.** *Let α be an algebraic number of degree d. Then if α is a kth root of unity, $k \leq 2d^2$.*

Proof. The degree of the kth root of unity is precisely $\Phi(k)$, where Φ denotes the Euler totient function. The desired inequality follows from the property that $\Phi(k) \geq \sqrt{k/2}$. ◀

We record a number-theoretic fact which describes the density of the integer multiples of an irrational x modulo 1 in the unit interval: its proof relies on continued fraction expansions and the Ostrowski numeration system [8, 7], and is deferred to Appendix B. This result is decisive when considering Non-Uniform Robustness.

▶ **Lemma 8.** *For every irrational number x, strictly decreasing real positive function ψ, and interval $\mathcal{I} = [a, b] \subset [0, 1]$, $a \neq b$, there exists $y_0 \in \mathcal{I}$ such that $[nx - y_0] < \psi(n)$ for infinitely many even n, and $y_1 \in \mathcal{I}$ such that $[nx - y_1] < \psi(n)$ for infinitely many odd n.*

The familiar density theorem is an immediate corollary of the above powerful result. Indeed, we can consider an interval of length $\varepsilon/2$, and take $\psi(n) = \varepsilon/2$.

▶ **Lemma 9.** *Let x be irrational, and $y \in [0, 1)$. For every $\varepsilon > 0$, there exist infinitely many even n, and infinitely many odd n such that $[nx - y] < \varepsilon$.*

The following application of the density theorem is central to the computation of the discrete $\mu(\mathbf{c}) = \liminf_{n\in\mathbb{N}} \langle \mathbf{p}, \mathbf{q_n} \rangle_{dom}$.

▶ **Lemma 10.** *Suppose θ is not a rational multiple of 2π. Let h_1, h_2 be real functions such that $h_1(t)$ is continuous with period 2π. Then*
$\liminf_n \left(h_1(n\theta) + h_2(-1^n)\right) = \min_{t\in[0,2\pi], b\in\{-1,1\}} \left(h_1(t) + h_2(b)\right)$.

We note that despite the observations and results mentioned in the preceding discussion, the Diophantine approximation type and Lagrange constant of most transcendental numbers are unknown. For instance, computing $L_\infty(\pi)$ is a longstanding and mathematically interesting open problem. We refer the reader to [24, Section 5] for a cursory survey of the history of relevant developments in the field of Diophantine approximation. This source reduces the computation of the constants $L(t)$ and $L_\infty(t)$ for transcendental numbers such as $t = \arcsin(3/5)/2\pi$ to the non-robust variants of Positivity problems for LRS of order 6. In §7, we prove analogous hardness results for robust Positivity problems of order 5. To that end, we define a similar classes of transcendental numbers relevant to our reduction. Let

$$\mathcal{A}_{alg} = \{p + iq \in \mathbb{C} \mid p, q \in \mathbb{A},\ p^2 + q^2 = 1, \forall n.\ (p + iq)^n \neq 1\}, \tag{11}$$

$$\mathcal{A}_{rat} = \{p + iq \in \mathbb{C} \mid p, q \in \mathbb{Q},\ p^2 + q^2 = 1, \forall n.\ (p + iq)^n \neq 1\}. \tag{12}$$

i.e., the set $\mathcal{A}_{alg}$ (resp. $\mathcal{A}_{rat}$) consists of algebraic (resp. rational) numbers on the unit circle in $\mathbb{C}$, none of which are roots of unity. In particular, writing $p + qi = e^{i2\pi\theta}$, we have that $\theta \notin \mathbb{Q}$. We denote:

$$\mathcal{T}_{alg} = \left\{t \in (-1/2, 1/2] \mid e^{2\pi it} \in \mathcal{A}_{alg}\right\};\ \mathcal{T}_{rat} = \left\{t \in (-1/2, 1/2] \mid e^{2\pi it} \in \mathcal{A}_{rat}\right\}. \tag{13}$$

The sets $\mathcal{T}_{alg}, \mathcal{T}_{rat}$ are dense in $(-\frac{1}{2}, \frac{1}{2}]$. We represent $t \in \mathcal{T}_{alg}$ (resp. $\mathcal{T}_{rat}$) by the real algebraic (resp. rational) numbers $(\cos 2\pi t, \sin 2\pi t)$. In general, we don't have a method to compute or approximate to arbitrary precision $L(t), L(t, s), L_\infty(t), L_\infty(t, s)$ given $s, t \in \mathcal{T}_{alg}$, or even $s, t \in \mathcal{T}_{rat}$.

▶ **Definition 11** (Number-theoretic hardness). *Let $\mathcal{T}$ denote one of $\mathcal{T}_{alg}, \mathcal{T}_{rat}$ as defined above. A decision problem is said to be $\mathcal{T}$-Diophantine hard (resp. $\mathcal{T}$-Lagrange hard), if its decidability entails that given any $s, t \in \mathcal{T}$ and rational $\varepsilon > 0$, one can compute rational ℓ_1, ℓ_2 such that: (a) $|\ell_1 - L(t)| < \varepsilon$ (resp. $|\ell_1 - L_\infty(t)| < \varepsilon$); (b) $|\ell_2 - L(t,s)| < \varepsilon$ (resp. $|\ell_2 - L_\infty(t,s)| < \varepsilon$).*

▶ **Theorem 12** (Main Hardness Result). *Problem 1 (resp. Problem 2) is $\mathcal{T}_{alg}$-Diophantine hard (resp. $\mathcal{T}_{alg}$-Lagrange hard) at order 5. For rational input, Problem 1 (resp. Problem 2) is $\mathcal{T}_{rat}$-Diophantine hard (resp. $\mathcal{T}_{rat}$-Lagrange hard) at order 5.*

As noted in [3], in view of the Lagrange hardness (Definition 11) of Ultimate Positivity at order 6 [24], Problem 3, which asks whether the given neighbourhood consists entirely of initialisations that produce an Ultimately Positive sequence, is also Lagrange hard at order 6. The idea is to use the existing reduction from the computation of Lagrange constants to Ultimate Positivity, and extend it to the robust variant: one simply constructs a neighbourhood of initialisations that has the hard instance of Ultimate Positivity on its surface, but otherwise lies entirely in the region where Ultimate Positivity is guaranteed.

▶ **Theorem 13.** *Problem 3 is $\mathcal{T}_{alg}$-Lagrange hard at order 6.*

5 Decidability of Uniform Robustness

In this section, we prove Theorem 2 by showing that we can implement Step 3 of the overview in §3.1: (constructively) decide whether there exists N_2 such that for all $n > N_2$,

$$v_n := (\mathbf{e_1}^T \mathbf{A}^n \mathbf{c})^2 - (\mathbf{e_1}^T \mathbf{A}^n \mathbf{g_1})^2 - \cdots - (\mathbf{e_1}^T \mathbf{A}^n \mathbf{g_4})^2 \geq 0. \tag{14}$$

▶ **Theorem 14** (First Main Decidability Result, restated). *Problem 2 (Robust Uniform Ultimate Positivity) is decidable for simple LRS. Problem 1 (Robust Positivity) is decidable for simple LRS up to order 5. Problems 1 and 2 are decidable for general LRS up to order 4.*

5.1 Simple LRS

We begin by treating simple LRS. The goal is to show that the current state of the art is equipped to handle instances relevant to this setting. Recall the discussion on the point-wise sums of products of simple LRS, surrounding equation 2. If the original LRS is simple, then inequality 14 is also an instance of Ultimate Positivity for simple LRS; indeed, its input can be seen to be real algebraic. In case we are only interested in Robust Ultimate Positivity, the non-constructive decision procedure [25] suffices, because it completely solves Ultimate Positivity for simple LRS.

As a corollary of the proof of the decidability of Positivity of simple LRS up to order 9 [23], Ultimate Positivity for simple LRS is *constructively* decidable if one of the following holds: **(a)** all characteristic roots have the same modulus; **(b)** there are at most three pairs of complex conjugates among the dominant (maximal modulus) characteristic roots.

We argue that for the original simple LRS $(u_n)_n$, 5 is the highest order that guarantees that at least one of the conditions holds for the resulting simple LRS $(v_n)_n$ in inequality 14. For this, we recall the property discussed after equation 2: if U is the set of characteristic roots of $(u_n)_n$, then the set of characteristic roots of $v_n = u_n^2$ is $V = \{\lambda_1\lambda_2 : \lambda_1, \lambda_2 \in U\}$. By Proposition 1, U contains a real positive dominant root ρ. It is clear that the dominant roots of V result from, and only from multiplying together pairs of dominant roots from U. If U does not have non-real dominant roots, neither does V. If $\lambda, \bar{\lambda} \in U$ are dominant non-real roots, then $\lambda\bar{\lambda} = \rho^2 \in \mathbb{R}$. If $U = \{\rho, \lambda_1, \lambda_2, \bar{\lambda_1}, \bar{\lambda_2}\}$, all dominant, then all roots of V are

dominant, and condition **(a)** is met. The only remaining case is that U has one pair of complex conjugates among its dominant roots: the scenario that results in most dominant roots in V is $U_{dom} = \{\rho, -\rho, \lambda, \bar{\lambda}\}$. Then, the dominant roots in V are $\{\rho^2, -\rho^2, \lambda^2, \bar{\lambda}^2, \pm\rho\lambda, \pm\rho\bar{\lambda}\}$: three conjugate pairs, and condition **(b)** is met. Finally, we record that order 5 is maximal: consider $U = \{\rho, \lambda_1, \lambda_2, \bar{\lambda_1}, \bar{\lambda_2}, \alpha\}$ with α non-dominant. Then V has five pairs of complex conjugates among its dominant roots, along with the presence of non-dominant roots.

5.2 Non-simple LRS

We treat order 4 LRS: our techniques naturally apply to lower orders too. We make extensive use of the real exponential polynomial closed form 4 and the surrounding discussion. The key lies in observing that in critical inequality 14, each squared term satisfies the same recurrence, and hence we can express it as

$$\langle \mathbf{p}, \mathbf{q_n} \rangle^2 - \langle \mathbf{b_1}, \mathbf{q_n} \rangle^2 - \cdots - \langle \mathbf{b_4}, \mathbf{q_n} \rangle^2 \geq 0 \quad \Leftrightarrow \quad \langle \mathbf{x}, \mathbf{r_n} \rangle \geq 0. \tag{15}$$

We choose $\{\mathbf{q_n}\}_{n \in \mathbb{N}}$ judiciously, expand the squares, use trigonometric identities, and collect the terms in $\mathbf{r_n}$ to obtain a new LRS. If all the characteristic roots of the original LRS are real, then $\mathbf{q_n}$ is free of trigonometric terms, and hence so is $\mathbf{r_n}$. Thus $\langle \mathbf{x}, \mathbf{r_n} \rangle$ is also an LRS with all real characteristic roots, and constructively deciding the existence of N_2 is easily done through elementary growth arguments. We shall thus assume the presence of a pair of complex conjugates among the characteristic roots. As discussed through Proposition 1, any decision regarding Ultimate Positivity is NO in the absence of a real positive dominant root. At order 4, this means that there is **exactly one pair of complex conjugates** among the roots. We further assume, without loss of generality, that **the real positive dominant root is unity**. We shall also assume **non-degeneracy**, i.e. the ratio of any pair of distinct roots of the characteristic polynomial is not a root of unity. This can be detected, courtesy Lemma 7. In our restricted setting, degeneracy can arise because: (a) -1 is a characteristic root; (b) a characteristic root is of the form $\rho e^{2\pi i \cdot \frac{\ell}{k}}$, i.e. a scaled k^{th} root of unity. In this case, any LRS $\langle \mathbf{v}, \mathbf{q_n} \rangle$ with roots $\{1, \alpha, \rho e^{\pm 2\pi i \cdot \frac{\ell}{k}}\}$ can be decomposed as the interleaving of $2k$ real LRS $\{u_n^{(i)}\}_{i=0}^{2k-1}$ where $u_n^{(i)} = \langle \mathbf{v}, \mathbf{q_{2nk+i}} \rangle$. Each of these LRS has characteristic roots $\{1, \rho^{2k}\} \cup \{\alpha^{2k}\}$.

The only possibility, therefore, is that the characteristic roots are $1, 1, \gamma, \bar{\gamma}$. Let $0 < |\gamma| = \rho \leq 1$, where $\gamma = \rho e^{i\theta}$ is not a scaled root of unity. We take inequality 15 as the starting point for our computations. Let $\mathbf{q_n} = \begin{bmatrix} n & 1 & \rho^n \cos(n\theta - \varphi) & \rho^n \sin(n\theta - \varphi) \end{bmatrix}^T$. Let $\mathbf{u_1}^T, \ldots, \mathbf{u_4}^T$ be the rows of the **invertible** matrix $\begin{bmatrix} \mathbf{b_1} & \ldots & \mathbf{b_4} \end{bmatrix}$. The table below shows the terms and coefficients on simplifying inequality 15.

Term of $\mathbf{r_n}$	**Coefficient** in $\mathbf{x}$	**Explicitly**
n^2	z_2	$p_1^2 - \langle \mathbf{u_1}, \mathbf{u_1} \rangle$
n	z_1	$2p_1p_2 - 2\langle \mathbf{u_1}, \mathbf{u_2} \rangle$
1	z_0	$p_2^2 - \langle \mathbf{u_2}, \mathbf{u_2} \rangle$
$n\rho^n \cos(n\theta - \varphi)$	x_2	$2p_1p_3 - 2\langle \mathbf{u_1}, \mathbf{u_3} \rangle$
$n\rho^n \sin(n\theta - \varphi)$	y_2	$2p_1p_4 - 2\langle \mathbf{u_1}, \mathbf{u_4} \rangle$
$\rho^n \cos(n\theta - \varphi)$	x_1	$2p_2p_3 - 2\langle \mathbf{u_2}, \mathbf{u_3} \rangle$
$\rho^n \sin(n\theta - \varphi)$	y_1	$2p_2p_4 - 2\langle \mathbf{u_2}, \mathbf{u_4} \rangle$
ρ^{2n}	w	$\frac{1}{2}(p_3^2 + p_4^2) - \frac{1}{2}(\langle \mathbf{u_3}, \mathbf{u_3} \rangle + \langle \mathbf{u_4}, \mathbf{u_4} \rangle)$
$\rho^{2n} \cos(2n\theta - 2\varphi)$	x_0	$\frac{1}{2}(p_3^2 - p_4^2) - \frac{1}{2}(\langle \mathbf{u_3}, \mathbf{u_3} \rangle - \langle \mathbf{u_4}, \mathbf{u_4} \rangle)$
$\rho^{2n} \sin(2n\theta - 2\varphi)$	y_0	$p_3p_4 - \langle \mathbf{u_3}, \mathbf{u_4} \rangle$

If $\rho < 1$, then the dominant growth rate for the problem to be non-trivial is $n^2, n, 1$, or ρ^{2n}: indeed, if the growth rate is ρ^n or $n\rho^n$, the leading terms would all be trigonometric, and the sign of the sequence oscillates. The former cases can be solved with straightforward growth arguments, while the last case results in an order 3 LRS that can easily be dealt with [23, 24]. We thus assume $\rho = 1$. Again, if $z_2 \neq 0$, then decidability is trivial because the dominant growth rate of n^2 is dictated by a single term; hence we assume $z_2 = 0$. In this case, there are two groups of terms, based on growth rate: one with n, the other with 1. To study these groups, we define

$$f(t) = z_1 + x_2 \cos(t - \varphi) + y_2 \sin(t - \varphi), \tag{16}$$

$$g(t) = z_0 + w + x_1 \cos(t - \varphi) + y_1 \sin(t - \varphi) + x_0 \cos(2t - 2\varphi) + y_0 \sin(2t - 2\varphi). \tag{17}$$

Since θ is not a rational multiple of 2π, $\{n\theta \bmod 2\pi\}$ is dense in $[0, 2\pi]$, and we invoke Lemma 10 to deduce

$$\liminf_{n\in\mathbb{N}} f(n\theta) = \min_{t\in[0,2\pi]} f(t) = z_1 - \sqrt{x_2^2 + y_2^2} = \mu. \tag{18}$$

If $\mu < 0$, then the critical inequality $nf(n\theta) + g(n\theta) \geq 0$ will be violated infinitely often. If $\mu > 0$, we can compute an N_2 beyond which it is guaranteed to be satisfied. We thus concern ourselves with the case where $\mu = 0$. Recall the discussion around μ when its concept was first defined after Proposition 1: it is an intrinsic property of the problem itself, and invariant under the "phase" φ chosen in the basis of solutions. We thus assume that φ is chosen in such a way that the minimum is attained at φ, i.e. $f(\varphi) = 0$. This choice can be made by applying the trigonometric identity $\cos(a - b) = \cos a \cos b + \sin a \sin b$ to $f(t)$. This means that $y_2 = 0$, and we choose $-z_1 = x_2 < 0$.

Now, if $g(\varphi) > 0$, we compute a positive lower bound on $f(t)$ for t such that $g(t) < 0$. This then results in an N_2 beyond which $nf(n\theta) + g(n\theta) \geq 0$ is guaranteed. If $g(\varphi) < 0$, then the inequality has infinitely many violations. This is due to Lemma 6, which asserts that there are infinitely many n for which $f(n\theta) \leq 2\pi^2 z_1/5n^2$. These n are necessarily such that $n\theta$ is close to φ, and the negativity of $g(n\theta)$ is thus decisive.

The final case that remains is $g(\varphi) = 0$. We argue that remarkably, it does not arise at all!

▶ **Lemma 15.** *If $z_2 = \mu = 0$, it cannot be the case that $f(\varphi) = g(\varphi) = 0$.*

Proof. Suppose, for the sake of contradiction, the scenario actually occurs. This means that $z_2 = z_1 + x_2 = z_0 + w + x_1 + x_0 = 0$. From the table, these respectively imply

$$\begin{aligned} p_1^2 &= \langle \mathbf{u_1}, \mathbf{u_1} \rangle, \\ p_1(p_2 + p_3) &= \langle \mathbf{u_1}, \mathbf{u_2} + \mathbf{u_3} \rangle, \\ (p_2 + p_3)^2 &= \langle \mathbf{u_2} + \mathbf{u_3}, \mathbf{u_2} + \mathbf{u_3} \rangle. \end{aligned}$$

This implies that $|\langle \mathbf{u_1}, \mathbf{u_2} + \mathbf{u_3} \rangle| = ||\mathbf{u_1}|| \cdot ||\mathbf{u_2} + \mathbf{u_3}||$, i.e. $\mathbf{u_1}$ is a scaled multiple of $\mathbf{u_2} + \mathbf{u_3}$. This contradicts the fact that the rows of the invertible $\begin{bmatrix} \mathbf{b_1} & \dots & \mathbf{b_4} \end{bmatrix}$ are linearly independent, and we are done. ◀

6 Non-uniform Robustness: Decidability at order four

In this section, we prove Theorem 3. The techniques naturally apply to lower orders, and we omit their explicit treatment. Recall the critical condition from our overview in §3.2:

$$\mu(\mathbf{c}') = \liminf_{n\in\mathbb{N}} \langle \mathbf{p}', \mathbf{q_n} \rangle_{dom} \geq 0 \tag{19}$$

for all $\mathbf{c}'$ in the neighbourhood is necessary for the decision to be YES; the inequality holding strictly is sufficient. Critical cases arise when the surface of the neighbourhood touches the region where $\mu = 0$, and the non-dominant terms, if any, can potentially have a negative contribution. We demonstrate that these can be detected and dealt with.

Since Proposition 1 guarantees the existence of a real positive dominant term, $\langle \mathbf{p}, \mathbf{q_n} \rangle_{dom}$ can only be of one of the following forms: **(I)** z; **(II)** $z + w(-1)^n$; **(III)** $z + x\cos n\theta + y \sin n\theta$; **(IV)** $z + x\cos n\theta + y\sin n\theta + w(-1)^n$, where x, y, z, w are linear in the initialisation $\mathbf{c}$. Cases (I), (II), and (III), (IV) where θ is a rational multiple of 2π (detected with Lemma 7) are the easiest. The region $\mu \geq 0$ is carved out by *finitely* many halfspaces, defined by separating hyperplanes of the form $z + bw + c_0 x + s_0 y = 0$. By elementary linear algebra and co-ordinate geometry (e.g. by working in a basis where the neighbourhood is a perfect hypersphere), one can determine whether $\mu > 0$ for the entire neighbourhood, or whether $\mu < 0$ for some points in the neighbourhood, or whether the neighbourhood touches a hyperplane. Each hyperplane has at most one point of tangency, whose algebraic coordinates can be solved for. These critical points are low-dimensional instances of Ultimate Positivity, and can be decided with the state of the art [25].

We therefore assume that θ is not a rational multiple of 2π, and we are in Case (III) or (IV). We apply Lemma 10, we get that

$$\mu(\mathbf{c}) = \liminf_{n \in \mathbb{N}} \langle \mathbf{p}, \mathbf{q_n} \rangle_{dom} = \min_{t \in \mathbb{R}, b \in \{\pm 1\}} z + x\cos t + y \sin t + wb = z - \sqrt{x^2 + y^2} - |w|. \quad (20)$$

If we are in Case (IV), there are no non-dominant roots, and $\mu \geq 0$ throughout the neighbourhood is necessary as well as sufficient for the decision to be YES. This is an algebraic condition, and can be checked using the First Order Theory of the Reals.[2]

Case (III) remains. $\langle \mathbf{q_n}, \mathbf{p} \rangle = z + x\cos n\theta + y \sin n\theta + w\alpha^n$, where $0 < |\alpha| < 1$. As discussed, we can use the First Order Theory of the Reals to check the sufficient $\mu > 0$, and the necessary $\mu \geq 0$ throughout the neighbourhood. We consider the scenario where the necessity check succeeds, but the sufficiency check fails. The decision can be NO only if there are points on the surface of the neighbourhood where $\mu = 0$, and the non-dominant $w\alpha^n$ can make a negative contribution. We describe how these points are found and analysed. First, we observe that the region $\mu \geq 0$ is given by the cone $z - \sqrt{x^2 + y^2} \geq 0$. It can be intuited as being carved out by a continuum of hyperplanes $z + x\cos\phi + y\sin\phi = 0$. We encode the above discussion to find the critical points with the following first order formula with free variable c, which stands for $\cos\phi$

$$\chi_1(c) := \exists s \exists \mathbf{c}'.\ (\mathbf{c}' - \mathbf{c})^T \mathbf{S} (\mathbf{c}' - \mathbf{c}) = 1 \wedge z' + cx' + sy' = 0 \wedge c^2 + s^2 = 1 \wedge w' \sim 0. \quad (21)$$

In the above $\sim$ is $\neq$ if the non-dominant root $\alpha < 0$, and is $<$ if $\alpha > 0$. We can use Theorem 17 to get an equivalent quantifier free formula: this comprises purely of polynomial (in-)equalities in the free variable c. The set of c, and hence $\cos\phi$, satisfying these, consists of finitely many intervals. Of course, Ultimate Positivity is guaranteed when this set is empty: it means there are no points threatening to violate Ultimate Positivity.

---

[2] Given input $\mathbf{a}, \mathbf{c}, \mathbf{S}$, we can choose $\mathbf{V}$ (cf equation 4). Our formula is

$$\forall \mathbf{c}'.\ (\mathbf{c}' - \mathbf{c})^T \mathbf{S} (\mathbf{c}' - \mathbf{c}) \leq 1 \Rightarrow \Bigg(\exists z', x', y', w', r_1, r_2.\ \left(\begin{bmatrix} z' & x' & y' & w' \end{bmatrix}^T = \mathbf{V}^{-1} \mathbf{c}' \right) \wedge (r_1, r_2 \geq 0)$$
$$\wedge (z' - r_1 - r_2 \geq 0) \wedge (r_1^2 = w'^2) \wedge (r_2^2 = x'^2 + y'^2) \Bigg).$$

We first dispose of the case where all intervals consist of single points. Consider an interval $\{c_0\}$ consisting of a single point. This is illustrated by the case of the ball touching the cone in Figure 1. Due to its origins and discrete occurrence, c_0 must be a root of a polynomial obtained by quantifier elimination on χ_1, and is hence algebraic. The corresponding critical point is the point of tangency of the neighbourhood with a hyperplane with a real algebraic equation. Thus, it generates a real algebraic instance of Ultimate Positivity, which can be decided with the techniques of [25].

If, however, the set of c satisfying χ_1 consists of intervals that have more than one point, then the techniques of [25] to decide Ultimate Positivity for a single point with algebraic coordinates are no longer accessible, in fact, the decision will always be NO in our setting. This situation is illustrated by the case of the ball nestled in cone in Figure 1. Let $[\phi_1, \phi_2]$ be an interval of ϕ such that: (i) all values of c between $\cos\phi_1$ and $\cos\phi_2$ satisfy χ_1, (ii) The corresponding witnesses z' are at most z_0, and (iii) The corresponding witnesses w' have magnitude at least some fixed w_0. In order to ensure robust Ultimate Positivity, we must have for each ϕ (and corresponding point on the cone with coordinates $z'(c), x' = -cz', y' = -z'\sin\phi, w'$) in this interval, the following inequality is violated only finitely often:

$$\langle \mathbf{p}', \mathbf{q_n} \rangle = z' - z'\cos n\theta\cos\phi - z'\sin n\theta\sin\phi + w\alpha^n = z' - z'\cos(n\theta - \phi) + w\alpha^n \geq 0. \quad (22)$$

We consider an even weaker inequality (courtesy Lemma 6), which, in this context, we argue is bound to be violated infinitely often:

$$z_0[n\theta - \phi]_{2\pi}^2 \geq 2w_0\alpha^n. \quad (23)$$

The argument hinges on Lemma 8, which we restate:

▶ **Lemma 16.** *For every irrational number x, strictly decreasing real positive function ψ, and interval $\mathcal{I} = [a, b] \subset [0, 1]$, $a \neq b$, there exists $y_0 \in \mathcal{I}$ such that $[nx - y_0] < \psi(n)$ for infinitely many even n, and $y_1 \in \mathcal{I}$ such that $[nx - y_1] < \psi(n)$ for infinitely many odd n.*

Now, if $\alpha < 0$, we use Lemma 16 on the irrational $\theta/2\pi$, and the decreasing $\sqrt{\frac{w_0|\alpha|^n}{2\pi^2 z_0}}$ to argue that there exists a ϕ in the desired interval, such that the weaker inequality will be violated for infinitely many n of the appropriate parity. The case $\alpha > 0$ is even simpler, as the parity does not matter. Thus, we can return NO if we are in the case where the set of c satisfying χ_1 (equation 21) consists of intervals that contain more than a single point.

7 Uniform Robustness: Hardness at order five

We shall prove Theorem 12 in this section. That is, given $s, t \in \mathcal{T}$ as defined in equation 13, we shall give $\mathbf{a}, \mathbf{c}, \mathbf{S}$ (real algebraic or rational as appropriate) such that varying $\mathbf{S} = \mathbf{G}^T\mathbf{G}$ while invoking $\mathbf{S}$-Robust Positivity decision procedures will enable us to approximate $L(t), L(t, s)$ and $L_\infty, L_\infty(t, s)$ to arbitrary precision.

We recall $t \in \mathcal{T}$ is specified by $(\cos 2\pi t, \sin 2\pi t)$. This tuple is real algebraic for $t \in \mathcal{T}_{alg}$, and rational for $t \in \mathcal{T}_{rat}$. If we wish to approximate the homogenous constants, we specify $s = 0$ with the representation $(1, 0)$. Our linear recurrence relation $\mathbf{a}$ is such that the roots of the characteristic polynomial are $1, 1, 1, e^{2\pi it}, e^{-2\pi it}$, i.e. the characteristic polynomial is $(X - 1)^3(X^2 - 2X\cos 2\pi t + 1)$.

Recalling the discussion surrounding equation 4, $u_n = \mathbf{e_1}^T\mathbf{A}^n\mathbf{c} = \langle \mathbf{p}, \mathbf{q_n} \rangle$. For the problem instance we create in our reduction, we choose rational $\mathbf{p} = \begin{bmatrix} r & 0 & 1 + \frac{r}{2} & -1 & 0 \end{bmatrix}^T$, where r is a parameter we use to tune our guess for $L(t, s)$ and $L_\infty(t, s)$; we choose our basis of closed

form solutions such that $\mathbf{q_n}^T = \mathbf{e_1}^T\mathbf{A}^n\mathbf{V} = \begin{bmatrix} n^2 & n & 1 & \cos(2\pi(nt-s)) & \sin(2\pi(nt-s)) \end{bmatrix}$. We choose $\mathbf{S} = \mathbf{G}^T\mathbf{G}/r^2$, where $\mathbf{G} = \mathbf{V}^{-1}$. Observe that depending on whether s, t are in $\mathcal{T}_{alg}$ or $\mathcal{T}_{rat}$, the input $\mathbf{a}, \mathbf{c}, \mathbf{S}$ is constructed to be real algebraic or rational, as desired. Our critical inequality is

$$\langle \mathbf{p}, \mathbf{q_n} \rangle = \mathbf{e_1}^T\mathbf{A}^n\mathbf{c} \geq \max_{d \in \mathcal{B}_\mathbf{S}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{d} = ||r(\mathbf{e_1}^T\mathbf{A}^n\mathbf{G}^{-1})^T|| = r||\mathbf{q_n}||. \tag{24}$$

Let $\Psi(n, r)$ denote the proposition $\langle \mathbf{p}, \mathbf{q_n} \rangle \geq r||\mathbf{q_n}||$. Our reduction works by proving that for any guess $r > 0$, given $\varepsilon > 0$, we can compute an N such that for all $n \geq N$

$$\Psi(n, r) \Rightarrow n[nt - s] > \frac{(1-\varepsilon)\sqrt{7r}}{4\pi}. \tag{25}$$

$$\neg\Psi(n, r) \Rightarrow n[nt - s] < \frac{\sqrt{7r}}{(1-\varepsilon)4\pi}. \tag{26}$$

To compute $L_\infty(t, s) = \liminf_{n \in \mathbb{N}} n[nt - s]$ by increasingly precise approximations, we query Robust Uniform Ultimate Positivity: does $\Psi(n, r)$ hold for all but finitely many n? If the decision is YES, then we use property 25 to argue that for any ε, $n[nt - s]$ exceeds $\frac{(1-\varepsilon)\sqrt{7r}}{4\pi}$ for all but finitely many n, hence $L_\infty(t, s)$ must be at least $\frac{\sqrt{7r}}{4\pi}$. Conversely, if the decision is NO, we use property 26 to deduce that for any ε, $n[nt - s]$ falls short of $\frac{\sqrt{7r}}{(1-\varepsilon)4\pi}$ for infinitely many n, hence $L_\infty(t, s)$ must be at most $\frac{\sqrt{7r}}{4\pi}$.

By definition, $L(t, s) = \inf_{n \in \mathbb{N}_{>0}} n[nt - s]$. Given the guess r, precision ε, the corresponding N, and oracle access to whether $\Psi(n, r)$ holds for all $n \geq N$, it follows from properties 25 and 26 that we can detect the truth of one of the cases among $\inf_{n \geq N} n[nt - s] \geq \frac{(1-\varepsilon)\sqrt{7r}}{4\pi}$ and $\inf_{n \geq N} n[nt - s] \leq \frac{\sqrt{7r}}{(1-\varepsilon)4\pi}$: indeed, the implications allow us to identify one from the set of valid cases, which is guaranteed to be non-empty. By obtaining sufficiently precise numerical approximations of $n[nt - s]$ for each $n < N$ in the finite prefix, one has a procedure for approximating $L(t, s)$.

In order to establish the number-theoretic hardness of Robust Positivity, we now explain how we use it as an oracle to decide whether $\Psi(n, r)$ holds for all $n \geq N$. Note that as it is, our query specifies a recurrence $\mathbf{a}$, an initialisation $\mathbf{c}$, a neighbourhood defined by $\mathbf{S}$ asks for the Robust Positivity of a *suffix* of the sequence, as opposed to the entire sequence. We create a new instance with updated $\mathbf{c}'$ and $\mathbf{S}'$ to implement the shift:

$$\forall n \geq N.\ \mathbf{e_1}^T\mathbf{A}^n\mathbf{c} \geq \max_{\mathbf{d} \in \mathcal{B}_\mathbf{S}} \mathbf{e_1}^T\mathbf{A}^n\mathbf{d} \ \Leftrightarrow\ \forall n.\ \mathbf{e_1}^T\mathbf{A}^n(\mathbf{A}^N\mathbf{c}) \geq \max_{\mathbf{d} \in \mathcal{B}_\mathbf{S}} \mathbf{e_1}^T\mathbf{A}^n(\mathbf{A}^N\mathbf{d}) \tag{27}$$

$$\Leftrightarrow\ \forall n.\ \mathbf{e_1}^T\mathbf{A}^n(\mathbf{c}') \geq \max_{\mathbf{d}' \in \mathcal{B}_{\mathbf{S}'}} \mathbf{e_1}^T\mathbf{A}^n(\mathbf{d}'). \tag{28}$$

It is clear that $\mathbf{c}' = \mathbf{A}^N\mathbf{c}$ and that $\mathbf{d}' = \mathbf{A}^N\mathbf{d}$. Using the same reasoning as we did in the derivation of equation 6, we argue $\mathbf{S}' = (\mathbf{A}^{-N})^T\mathbf{S}\mathbf{A}^{-N}$. The reduction is thus complete, but for the proof of properties 25 and 26, which we defer to Appendix C.

8 Extensions and Perspective

It is interesting to investigate what kinds of decidability and hardness results hold for neighbourhoods specified using norms other than those based on the standard matrix inner product. For instance, our techniques for $\mathbf{S}$-Robust Non-uniform Ultimate Positivity hinged on the First Order Theory of the Reals and were rather agnostic to the exact shape of the neighbourhood: we can easily extend the same techniques to arbitrary *semi-algebraic*

neighbourhoods. Perhaps, other results could also be universal across a wider class of norms, and there could be a profound underlying linear-algebraic reason whose discovery would be mathematically significant.

While contributing towards a sharp and comprehensive picture of what is *decidable* about Robust Positivity for LRS, we found it remarkable that number-theoretic analyses involving Diophantine approximation, which usually show up in the context of hardness, also play a significant role in our *decidability* proofs! However, a rather conspicuous gap in our picture is the status of **S**-Robust Non-uniform Ultimate Positivity at order 5: this seems to require even more delicate analysis.

An obvious, but possibly tedious future direction would be to tie up the book-keeping loose ends, and meticulously account for the complexity of our techniques. We chose to work with algebraic numbers; in settings involving rational numbers where scaling to integers and accessing an PosSLP oracle is viable, the complexity usually lies in PSPACE. However, this might blow up significantly in the absence of efficient positivity testing for a different class of arithmetic circuit.

In the grand Formal Methods scheme, the study of Hyperproperties [10] is an exciting natural way robustness problems for Linear Dynamical Systems could fit in. Hyperlogics reason about sets of traces of an infinite time system, rather than a single trace. They gained importance as a means to verify security in view of attacks like Meltdown and Spectre. A quintessential hyperproperty, for instance, would specify a reasonable notion of *indistinguishability* of traces. In that regard, our notions of **S**-Robust Positivity and **S**-Robust Uniform Ultimate Positivity bear striking resemblance. Exploring deeper connections is a fascinating future research avenue.

References

1. S. Akshay, Nikhil Balaji, Aniket Murhekar, Rohith Varma, and Nikhil Vyas. Near-Optimal Complexity Bounds for Fragments of the Skolem Problem. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2020.37.
2. S. Akshay, Hugo Bazille, Blaise Genest, and Mihir Vahanwala. On robustness for the skolem and positivity problems. In *STACS 2022*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.STACS.2022.5.
3. S. Akshay, Hugo Bazille, Blaise Genest, and Mihir Vahanwala. On robustness for the skolem, positivity and ultimate positivity problems, 2022. arXiv:2211.02365.
4. S. Akshay, Blaise Genest, and Nikhil Vyas. Distribution-based objectives for Markov Decision Processes. In *33rd Symposium on Logic in Computer Science (LICS 2018)*, volume IEEE, pages 36–45, 2018.
5. Christel Baier, Florian Funke, Simon Jantsch, Toghrul Karimov, Engel Lefaucheux, Joël Ouaknine, Amaury Pouly, David Purser, and Markus A. Whiteland. Reachability in dynamical systems with rounding. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020*, volume 182 of *LIPIcs*, pages 36:1–36:17, 2020.
6. S. Basu, R. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2nd edition, 2006.
7. Valérie Berthé and Jungwon Lee. Dynamics of ostrowski skew-product: I. limit laws and hausdorff dimensions, 2022. arXiv:2108.06780.
8. Avraham Bourla. The ostrowski expansions revealed, 2016. arXiv:1605.07992.

9 Mark Braverman. Termination of integer linear programs. In *International Conference on Computer Aided Verification*, pages 372–385. Springer, 2006.

10 Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *2008 21st IEEE Computer Security Foundations Symposium*, pages 51–65, 2008. doi:10.1109/CSF.2008.7.

11 H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1993.

12 Julian D'Costa, Toghrul Karimov, Rupak Majumdar, Joël Ouaknine, Mahmoud Salamati, Sadegh Soudjani, and James Worrell. The pseudo-Skolem problem is decidable. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021*, volume 202 of *LIPIcs*, pages 34:1–34:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

13 Julian D'Costa, Toghrul Karimov, Rupak Majumdar, Joël Ouaknine, Mahmoud Salamati, and James Worrell. The Pseudo-Reachability Problem for Diagonalisable Linear Dynamical Systems. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:13, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.40.

14 Manuel Eberl and René Thiemann. Factorization of polynomials with algebraic coefficients. *Archive of Formal Proofs*, November 2021. Formal proof development. URL: https://isa-afp.org/entries/Factor_Algebraic_Polynomial.html.

15 Graham Everest, Alfred J. van der Poorten, Igor E. Shparlinski, and Thomas Ward. Recurrence Sequences. In *Mathematical surveys and monographs*, 2003.

16 Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem's problem – On the border between decidability and undecidability. Technical Report 683, Turku Centre for Computer Science, 2005.

17 A Khintchine. Neuer beweis und verallgemeinerung eines hurwitzschen satzes. *Mathematische Annalen*, 111:631–637, 1935.

18 J. C. Lagarias and J. O. Shallit. Linear fractional transformations of continued fractions with bounded partial quotients. *Journal de théorie des nombres de Bordeaux*, 9:267–279, 1997.

19 Richard Lipton, Florian Luca, Joris Nieuwveld, Joël Ouaknine, David Purser, and James Worrell. On the skolem problem and the skolem conjecture. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533328.

20 M. Mignotte. Some useful bounds. In *Computer Algebra*, 1982.

21 Maurice Mignotte, Tarlok Nath Shorey, and Robert Tijdeman. The distance between terms of an algebraic recurrence sequence. *Journal für die reine und angewandte Mathematik*, 349:63–76, 1984.

22 Eike Neumann. Decision problems for linear recurrences involving arbitrary real numbers. *Logical Methods in Computer Science*, 17(3), 2021.

23 Joël Ouaknine and James Worrell. On the positivity problem for simple linear recurrence sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 318–329. Springer, 2014.

24 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. SIAM, 2014.

25 Joël Ouaknine and James Worrell. Ultimate positivity is decidable for simple linear recurrence sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 330–341. Springer, 2014.

26 James Renegar. On the Computational Complexity and Geometry of the First-Order Theory of the Reals, Part I: Introduction. Preliminaries. The Geometry of Semi-Algebraic Sets. The Decision Problem for the Existential Theory of the Reals. *J. Symb. Comput.*, 13:255–300, 1992.

27 Ashish Tiwari. Termination of linear programs. In *Computer-Aided Verification, CAV*, volume 3114 of *LNCS*, pages 70–82. Springer, July 2004.

28 Nikolai Vereshchagin. The problem of appearance of a zero in a linear recurrence sequence. *Mat. Zametki*, 38(2):609–615, 1985.

A Appendix: Notation and Prerequisites

For the purposes of discussing robustness, we shall use $\mathcal{B}$ to denote the unit Euclidean ball in $\mathbb{R}^\kappa$, centred at the origin. Similarly, we use $\mathcal{B}_\mathbf{S}$ to denote the set of $\mathbf{d}$ such that $\mathbf{d}^T\mathbf{S}\mathbf{d} \leq 1$. For real column vectors $\mathbf{x}, \mathbf{y}$, we use $\langle \mathbf{x}, \mathbf{y} \rangle$ to denote the inner product $\mathbf{x}^T\mathbf{y} = \mathbf{y}^T\mathbf{x}$. The notation $||\mathbf{x}||$ denotes the standard ℓ^2-norm $\sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

Throughout this paper, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ respectively denote the natural numbers, integers, rationals, reals, and complex numbers. $\alpha \in \mathbb{C}$ is said to be algebraic if it is a root of a polynomial with integer coefficients. Algebraic numbers form an algebraically closed field, denoted by $\overline{\mathbb{Q}}$. We denote the field of real algebraic numbers by $\mathbb{A}$.

This Appendix contains a brief initiation to this number field $\mathbb{A}$ and $\overline{\mathbb{Q}}$. The key takeaways are that the usual arithmetic as well as polynomial root computation can be carried out with perfect precision, and that the First Order Theory of the Reals $\langle \mathbb{R}; +, \cdot, \geq, 0, 1 \rangle$ is a decidable logical system powerful enough to fit our purposes.

A.1 Algebraic Numbers: Arithmetic

For an algebraic number α, its defining polynomial p_α is the unique polynomial in $\mathbb{Z}[X]$ of least degree such that the GCD of its coefficients is 1 and α is one of its roots. Given a polynomial $p \in \mathbb{Z}[X]$, we denote the length of its representation by $\text{size}(p)$; its height, denoted by $H(p)$, is the maximum absolute value of the coefficients of p; $d(p)$ denotes the degree of p. The height $H(\alpha)$ and degree $d(\alpha)$ of α are defined to be the height and degree of p_α.

For any $p \in \mathbb{Z}[X]$, the distance between distinct roots is effectively lower bounded in terms of its degree and height [20]. This bound allows one to represent an algebraic number α as a 4-tuple (p, a, b, r) where p is the defining polynomial, and $a + bi$ is a rational approximation of sufficient precision $r \in \mathbb{Q}$. We use $\text{size}(\alpha)$ to denote the size of this representation, i.e., number of bits needed to write down this 4-tuple.

Given a polynomial $p \in \mathbb{Z}[X]$, one can compute its roots in polynomial time [6]. Recently, implementations of algorithms to factor polynomials in $\overline{\mathbb{Q}}[X]$ have been verified [14]. Given α, β two algebraic numbers, one can always compute the representations of $\alpha + \beta$, $\alpha\beta$, $\frac{1}{\alpha}$, $\Re(\alpha)$, $\Im(\alpha)$, $|\alpha|$, and decide $\alpha = \beta$, $\alpha > \beta$ in polynomial time with respect to the size of their representations. [6, 11].

A.2 First Order Theory of the Reals

This logical theory reasons about the universe of real numbers, and is denoted $\langle \mathbb{R}; +, \cdot, \geq, 0, 1 \rangle$. That is, variables take real values; terms can be added and multiplied, we have the comparison predicate, and direct access to the constants 0 and 1. Thus, our propositional atoms are inequalities involving polynomials with integer coefficients. With existential quantifiers and polynomials, we can thus express algebraic constants too. Formally, we have access to only the existential quantifier, negation, and disjunction; however, this can express the universal quantifier and all other Boolean connectives as well.

Variables are either quantified or free. Remarkably, the First Order Theory of the Reals admits quantifier elimination: for any formula $\chi(\mathbf{x})$, whose free variables are $\mathbf{x}$, there exists an **equivalent** formula $\psi(\mathbf{x})$ that does not contain any quantified variables. The following result is relevant to us.

▶ **Theorem 17** (Renegar [26]). *Let $\chi(\mathbf{x})$ be a first order formula interpreted over the theory of the reals. There exists a procedure that returns an equivalent quantifier-free formula $\psi(\mathbf{x})$ in disjunctive normal form. Moreover, if the total number of variables in χ is bounded a priori, this procedure runs in time polynomial in the size of the representation of χ.*

B Appendix: Ostrowski Numeration System

In this appendix, we prove Lemma 8. We state number-theoretic properties of the continued fraction representation and Ostrowski Numeration System without proof. We refer the reader to [8] for a more detailed exposition, and we closely follow the discussion surrounding [7, Propositions 1.1, 2.1] in our own proof. We first prove a slightly simpler statement.

▶ **Lemma 18.** *For every irrational number x, strictly decreasing real positive function ψ, and interval $\mathcal{I} = [\alpha, \beta] \subset [0, 1]$, $\alpha \neq \beta$, there exists $y \in \mathcal{I}$ such that $[nx - y] < \psi(n)$ for infinitely many n.*

Proof. Without loss of generality, we can assume that $x \in (0, 1)$. Consider the continued fraction representation of x: $[0; a_1, a_2, a_3, \dots]$

$$x = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots}}}}$$

where $a_1, a_2, a_3, \dots \in \mathbb{N}$. Let the rational approximation of x obtained by truncating the expansion at the kth level be $\frac{p_k}{q_k}$, i.e. $\frac{p_1}{q_1} = \frac{1}{a_1}$, and so on. Let $\theta_k = q_k x - p_k$. We have that $|\theta_k| = (-1)^k \theta_k$. It is well known that $|\theta_k| < 1/q_k$. We define $q_{-1} = p_0 := 0$, and $p_{-1} = q_0 := 1$, so that for $k \geq 1$, the following recurrences hold:

$$p_k = a_k p_{k-1} + p_{k-2}, \;\; q_k = a_k q_{k-1} + q_{k-2}.$$

We thus have that $q_k \geq \left(\frac{1+\sqrt{5}}{2}\right)^k = \phi^k$.

▶ **Proposition 19** ([7]). *Let irrational x and its continued fraction representation $[0; a_1, a_2, a_3, \dots]$ be as above. The infinite series*

$$\sum_{i=1}^{\infty} a_i |\theta_{i-1}|$$

converges.

▶ **Proposition 20** (Ostrowski Numeration System, [7]). *Every real number $y \in [0, 1)$ can be written uniquely in the form*

$$y = \sum_{i=1}^{\infty} b_i |\theta_{i-1}| = \sum_{i=1}^{\infty} (-1)^{i-1} b_i \theta_{i-1}$$

where $b_i \in \mathbb{N}$ $b_i \leq a_i$ for all $i \geq 1$. If for some i, $a_i = b_i$, then $b_{i+1} = 0$. $a_i \neq b_i$ for infinitely many odd, and infinitely many even indices i.

We prove Lemma 18 by using the free choice of b_i in this system to construct appropriate y. We first handle the issue of placing y in the correct interval $[\alpha, \beta]$. Let $\beta - \alpha = \delta$. We use Proposition 19 to argue that there exists a suffix of the infinite series, such that changing the suffix does not change the real number it represents by more than $\delta/2$. Then, we can simply fix the corresponding prefix of $(\alpha + \beta)/2$ to be the prefix of y.

Once this prefix is locked in, our strategy is to set b_i to 0 in even positions, and 1 in some odd positions, to ensure that for sufficiently large k, $n_k = \sum_{i=1}^{k} b_i(-1)^{i-1}q_{i-1}$ is positive, and increasing in k.

Now, notice that since b_i, p_i are all integers, for any y,

$$\begin{aligned}
[n_k x - y] &= \left[\sum_{i=1}^{k} b_i(-1)^{i-1}q_{i-1}x - \sum_{i=1}^{k} b_i(-1)^{i-1}p_{i-1} - y\right] \\
&= \left[\sum_{i=1}^{k} b_i(-1)^{i-1}\theta_{i-1} - y\right] \\
&= \left[-\sum_{i=k+1}^{\infty} b_i(-1)^{i-1}\theta_{i-1}\right] = \sum_{i=k+1}^{\infty} b_i|\theta_{i-1}| \\
&< \sum_{i=k+1}^{\infty} b_i\frac{1}{q_{i-1}} \leq \sum_{i=k+1}^{\infty} b_i\frac{1}{\phi^{i-1}} \leq \frac{c}{\phi^k}.
\end{aligned}$$

Note that the last constant c can be set independently of the choice of which b_i are 1, and which are 0: it comes from the convergence of the geometric sum. We now make the choice of where to set $b_i = 1$. To conclude the proof, we shall show that given a decreasing function ψ, we can ensure that for infinitely many distinct n_k,

$$[n_k x - y] < \frac{c}{\phi^k} \leq \psi(n_k) = \psi\left(\sum_{i=1}^{k} b_i(-1)^{i-1}q_{i-1}\right).$$

The first inequality is guaranteed. Suppose the second inequality does not hold. Then, from $i = k$ onwards, we keep assigning $b_i := 0$. This holds n_k constant as k increases, but decreases $\frac{c}{\phi^k}$. Eventually, the second inequality will indeed hold. After this point, for the next odd i, we can set b_i to 1, and get a new n_k. We continue this ad infinitum, and we are done. ◀

Now, to get infinitely many *even* n, apply Lemma 18 with $2x$, $[a, b]$, $\psi_0(n) = \psi(2n)$. For some choice of y, there will be infinitely many n such that $[2nx - y] < \psi_0(n) = \psi(2n)$. To get infinitely many *odd* n, we can take a subset of the interval, and shift it by x. Take $\psi_1(n) = \psi(2n + 1)$. For some choice of $y - x$, there will be infinitely many n such that $[n(2x) - (y - x)] = [(2n + 1)x - y] < \psi_1(n) = \psi(2n + 1)$.

C Appendix: Technical details of hardness proof

In this Appendix, we prove properties 25 and 26, which we restate:

$$\Psi(n, r) \Rightarrow n[nt - s] > \frac{(1 - \varepsilon)\sqrt{7r}}{4\pi}. \tag{29}$$

$$\neg\Psi(n, r) \Rightarrow n[nt - s] < \frac{\sqrt{7r}}{(1 - \varepsilon)4\pi}. \tag{30}$$

By definition, $\Psi(n,r)$ holds if and only if $rn^2 + \frac{r}{2} + 1 - \cos 2(\pi(nt-s)) \geq r\sqrt{n^4+n^2+2}$. Through elementary algebraic manipulations, we can alternately group the terms as

$$1 - \cos(2\pi(nt-s)) \geq \frac{r}{2}\left(\frac{7n^2+14}{(n^2+\sqrt{n^4+n^2+2})(n^2+4+\sqrt{n^4+n^2+2})}\right) = r \cdot Q(n). \quad (31)$$

We note that in the limit, the ratio of $Q(n)$ to $7/8n^2$ tends to 1 from below. On the other hand, for small values of x, the expression $x^2/2$ is a close over-approximation for $1 - \cos x$. We capture the crucial interdependence in the following technical lemma.

▶ **Lemma 21.** *Let $r > 0$. For every $\varepsilon > 0$, we can compute N such that*

1. *For all $n \geq N$, $Q(n) > 7(1-\varepsilon)^2/8n^2$.*
2. $1 - \cos x < 7r/8N^2 \;\Rightarrow\; 1 - \cos x \geq (1-\varepsilon)^2 x^2/2$.

For some r, ε, let N be computed by Lemma 21. Consider $n \geq N$. In case $\Psi(n,r)$ holds, property 25 follows by considering the beginning and end of the chain of inequalities

$$2\pi^2[nt-s]^2 = \frac{[2\pi(nt-s)]_{2\pi}^2}{2} \geq 1 - \cos(2\pi(nt-s)) \geq r \cdot Q(n) > \frac{7r(1-\varepsilon)^2}{8n^2}. \quad (32)$$

Similarly, if $\neg\Psi(n,r)$ holds, we can use Lemma 21 to construct the chain

$$2\pi^2(1-\varepsilon)^2[nt-s]^2 = \frac{(1-\varepsilon)^2[2\pi(nt-s)]_{2\pi}^2}{2} \leq 1 - \cos(2\pi(nt-s)) < r \cdot Q(n) < \frac{7r}{8N^2}. \quad (33)$$

Nash Equilibria of Two-Player Matrix Games Repeated Until Collision

Aniket Murhekar
University of Illinois, Urbana-Champaign, IL, USA

Eklavya Sharma
University of Illinois, Urbana-Champaign, IL, USA

Abstract

We introduce and initiate the study of a natural class of repeated two-player matrix games, called *Repeated-Until-Collision* (RUC) games. In each round, both players simultaneously pick an action from a common action set $\{1, 2, \ldots, n\}$. Depending on their chosen actions, they derive payoffs given by $n \times n$ matrices A and B, respectively. If their actions *collide* (i.e., they pick the same action), the game ends, otherwise, it proceeds to the next round. Both players want to maximize their total payoff until the game ends. RUC games can be interpreted as pursuit-evasion games or repeated hide-and-seek games. They also generalize *hand cricket*, a popular game among children in India.

We show that under mild assumptions on the payoff matrices, every RUC game admits a Nash equilibrium (NE). Moreover, we show the existence of a *stationary* NE, where each player chooses their action according to a probability distribution over the action set that does not change across rounds. Remarkably, we show that all NE are effectively the same as the stationary NE, thus showing that RUC games admit an *almost* unique NE. Lastly, we also show how to compute (approximate) NE for RUC games.

2012 ACM Subject Classification Applied computing → Economics

Keywords and phrases Two player games, Nash equilibrium, Repeated games

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.18

Related Version *Full Version*: `https://arxiv.org/abs/2309.15870v1` [23]

Acknowledgements We thank Prof. Jugal Garg and Prof. Ruta Mehta for their helpful comments.

1 Introduction

Two-player matrix games, or *bimatrix* games, are among the most well-studied classes of games in non-cooperative game theory [27]. A bimatrix game involves two players with a finite set of actions $\{1, 2, \ldots, n\}$, and can be represented by two $n \times n$ payoff matrices A and B for the two players. A play of the game can be thought of as one player choosing a row i and the other player choosing a column j simultaneously. The "row" player gets a payoff of $A_{i,j}$ and the "column" player gets a payoff of $B_{i,j}$. To avoid being predictable, players can choose their actions according to a *randomized* or mixed strategy which is a probability distribution over their action set. Naturally, each player is interested in maximizing their (expected) payoff.

Arguably the most popular solution concept in game theory is that of Nash equilibrium [24]. A Nash equilibrium (NE) for a bimatrix game is a pair of mixed strategies where no player has any incentive to unilaterally deviate and change her strategy. In a celebrated result, Nash showed that each bimatrix game admits a Nash equilibrium. Given that two-player games are ubiquitous and have widespread applications in networks and communications [8], financial markets [1], robotics [18], etc., the field of algorithmic game theory has extensively studied the existence and computation of NE in bimatrix games and their generalizations.

© Aniket Murhekar and Eklavya Sharma;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 18; pp. 18:1–18:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A repeated two-player game is one such generalization, which involves repetitions of a base bimatrix game [20, 33, 6]. In a repeated game, a player can choose their moves based on the moves played by the players in the previous rounds, while recognizing that their current move will impact the choice of moves of their opponent in the future, leading to added complexity. While finitely repeated games are well-understood using backward-induction [6], the equilibria of infinitely repeated games can differ from that of the base game [20].

In this paper, we introduce and initiate the study of a natural class of repeated bimatrix games, called *Repeated-Until-Collision* (RUC) games. For $n \times n$ non-negative matrices A and B, an RUC game (A, B) has two players, called *max player* and *min player*, where A is called the max player's score matrix and B is called the min player's cost matrix. The game consists of multiple rounds. Suppose the max player and min player pick actions i and j, respectively, in a round. Then the max player earns a score of $A_{i,j}$ and the min player incurs a cost of $B_{i,j}$. If $i = j$, the game ends (we call this event a *collision*). Otherwise, the game proceeds to the next round. We assume that the max player and min player may use randomized strategies. The max player wants to maximize her expected total score and the min player wants to minimize her expected total cost. Below we discuss a few applications of RUC games, thus underscoring their importance.

Applications. RUC games can be interpreted as variants of pursuit-evasion games [17, 7]. As a simplified example, consider a game between a drug dealer (max player) and the law enforcement (min player). Each day, the drug dealer chooses one of n locations for a drop, while law enforcement chooses a location for a random check. If their locations coincide, the game ends as the drug dealer is caught. Until then, the drug dealer wants to maximize his revenue from the sale of drugs, while law enforcement wants to minimize the cost to society by illicit drug use. Similar examples can be found in reinforcement learning or robot motion planning, where an agent, e.g., a spy robot, is trying to learn an unknown environment. The agent gets a reward from exploring one of n locations, while its adversary (security systems, nature, etc.) actively tries to minimize how much the agent discovers before catching the infiltrating agent.

RUC games generalize *hand cricket* [13, 3], a popular game played by children in India. Hand cricket is a contest between a "batter" (max player) and a "bowler" (min player). In a round, the batter and bowler simultaneously choose actions i and j from $\{1, \ldots, n\}$. This gives the batter a score of i, while the bowler suffers a cost of i. The game ends if $i = j$, i.e., the batter is declared "out". Hand cricket involves the batter trying to maximize her total score and the bowler trying to minimize it. Note that this is an example of a *zero-sum* RUC game, where the payoff to the max player equals the cost borne by the min player.

We also note that RUC games can be thought of as repeated hide-and-seek games [31, 30] between two players, the Hider and the Seeker. In a hide-and-seek game, there are n locations containing varying rewards. The Hider tries to collect as much reward as possible, before getting caught by the Seeker, who aims to minimize the reward lost to the Hider. Like hand cricket, repeated hide-and-seek is also a zero-sum RUC game.

Nash equilibria in RUC games. Our paper addresses the following natural questions about RUC games:

> *Do Repeated-Until-Collision games admit Nash equilibria? If so, are they unique? Can they be computed efficiently?*

At first glance, it is not even clear if an NE should exist for general RUC games. Intuitively, the min player would like a collision to happen soon to prevent accumulating a large cost, while the max player would like to delay a collision. Additionally, if some row of A has large

numbers, the max player would want to pick the action corresponding to that row more frequently. These two approaches are at odds with each other, e.g., if the max player picks a single action very frequently, the min player can pick the same action and cause a collision very soon. Hence, the players must pick distributions of actions that balance the per-round score (or cost) and the duration of the game. Whether distributions in which neither player has an incentive to deviate unilaterally (i.e., Nash equilibrium) exist is unclear.

1.1 Our Results

As a warm-up, we begin by looking at a simpler version of RUC games called *stationary Repeated-Until-Collision* (SRUC) games, where the players are restricted to use only *stationary strategies*. A stationary strategy is one where in each round, the player samples an action from the same distribution. Formally, given a vector $\mathrm{x} \in \Delta_n$ (where Δ_n is the n-dimensional standard simplex), the stationary strategy x is to pick action i in each round independently randomly with probability x_i for each $i \in \{1, \ldots, n\}$. Intuitively, stationary strategies make sense for RUC games, since the game has a recursive structure: if a collision doesn't happen in the first round, the remaining game is identical to the original. Moreover, stationary strategies are a natural choice for players constrained on computational resources, since they are independent of the game history.

Under mild assumptions on matrices A and B, we show that a Nash equilibrium always exists for the SRUC game (A, B), and is unique.

▶ **Definition 1.** *For any matrix $A \in \mathbb{R}^{n \times n}_{\geq 0}$, let* $\mathrm{graph}(A) := (V, E)$ *be a directed graph where* $V := \{1, \ldots, n\}$ *and* $E := \{(i, j) : i \in V, j \in V \setminus \{i\}, A_{i,j} > 0\}$. *Then A is called* irreducible *iff* $\mathrm{graph}(A)$ *is strongly connected.*

▶ **Theorem 2.** *Let A and B be irreducible matrices. Then a Nash equilibrium exists for the SRUC game (A, B). Furthermore, the Nash equilibrium is unique iff* $\mathrm{graph}(A)$ *is a subgraph of* $\mathrm{graph}(B)$.

We prove Theorem 2 in Section 3. Our existence result uses the Perron-Frobenius theorem [21], a central result from matrix theory, and shows that the NE strategies can be computed from the leading eigenvectors of the payoff matrices.

In Appendix E of the full version [23], using more involved techniques, we also show the existence of NE for SRUC games with reducible payoff matrices. Furthermore, if either A or B is reducible, Nash equilibria need not be unique.

In Section 4, we switch back from SRUC games to RUC games, i.e., we allow players to play non-stationary strategies. This setting is significantly more challenging to analyze due to its large and complicated strategy space. Since strategies can be history-dependent, a deterministic strategy can be viewed as a function assigning an action to every possible state of the game, where the state is given by the players' actions in past rounds. Such functions have an infinite domain since the game's history can grow arbitrarily large, which implies that the set of deterministic strategies is uncountably infinite. So, randomized strategies, defined as probability distributions over deterministic strategies, are tough to analyze formally since these probability distributions have a large and unusual sample space.

Although RUC games allow non-stationary strategies, it is not obvious if players benefit from this extra freedom. If one player uses a stationary strategy, it is a priori unclear if the other player can gain by deviating unilaterally to a non-stationary strategy. Interestingly, we show that every RUC game admits a Nash equilibrium with stationary strategies.

▶ **Theorem 3.** *Let A and B be irreducible matrices. Then there exists a pair of stationary strategies $(\mathrm{x}^*, \mathrm{y}^*)$ that is a Nash equilibrium for both the RUC game (A, B) and the SRUC game (A, B).*

Furthermore, we show that for zero-sum RUC games (i.e., when $A = B$), the Nash equilibrium $(\mathrm{x}^*, \mathrm{y}^*)$ is *unique up to equivalence*, i.e., it is impossible to distinguish between different Nash equilibria just by observing the players' actions.

Having studied the existence of Nash equilibria in RUC games, we now turn to computation. Since the eigenvectors could be irrational, we consider *approximate* Nash equilibria. In an ε-approximate NE, neither player can improve their payoff by a factor of $(1 + \varepsilon)$ through unilateral deviations. We show that fine-enough approximations to the leading eigenvectors of the payoff matrices can be used to compute approximate NE of an RUC game.

▶ **Theorem 4.** *Let $(\mathrm{x}^*, \mathrm{y}^*)$ be a Nash equilibrium for the RUC game (A, B) with full support. For $\varepsilon \in [0, 1)$, let $\widehat{\mathrm{x}}, \widehat{\mathrm{y}} \in \Delta_n$ be such that $|\mathrm{x}^*_i - \widehat{\mathrm{x}}_i| \leq \varepsilon \mathrm{x}^*_i$ and $|\mathrm{y}^*_i - \widehat{\mathrm{y}}_i| \leq \varepsilon \mathrm{y}^*_i$, for all $i \in [1..n]$. Then $(\widehat{\mathrm{x}}, \widehat{\mathrm{y}})$ is a $\frac{4\varepsilon}{(1-\varepsilon)^2}$-approximate Nash equilibrium.*

Estimating the leading eigenvalue and eigenvector of a matrix can be done via the power method [22], an iterative method with a linear convergence rate.

1.2 Related Work

Among the earliest results on bimatrix games are von Neumann's Minimax Theorem in 1928 on zero-sum games [25], and Nash's fundamental result proving the existence of Nash equilibria in bimatrix games [24]. Subsequently the field of algorithmic game theory [26] has devoted considerable attention to the computation of NE in games, with a series of works showing PPAD-completeness [28] even for computing an approximate NE [28, 12, 9]. PPAD-hardness was then shown for many important subclasses, including constant-rank games [19], sparse games [10], win-lose games [11], etc. On the other hand, approximation schemes for finding approximate NE are known for classes like low rank games [16, 2] (FPTAS), and when $(A + B)$ is sparse [5] (PTAS).

Repeated games are well-understood in terms of "Folk Theorems", which indicate that several models of repeated games admit many Nash equilibria [14]. There are many models of repeated games, differing on the horizon for which the game is played (finite or infinite), aggregate utility to the players (arithmetic mean of payoffs in each round, or a sum of payoffs with a discount $\delta \in (0, 1)$), and the kind of equilibrium in consideration (Nash or subgame perfect Nash equilibrium). Finite games with arithmetic mean of payoffs as the utility admit a Nash equilibrium via backward induction [6]. Infinite games where the utility is the limit of the arithmetic mean of the payoffs have been shown to admit Nash equilibria assuming certain kinds of punishments used to deter players from deviating [4, 29]. Infinite games with the utility being a discounted sum of payoffs per round have also been shown to admit equilibria under different conditions on the base game and punishments [15, 14]. In contrast to these works, in our model of RUC games, the utility of a player is the (undiscounted) sum of per-round payoffs. Furthermore, while an RUC game can allow for infinitely many rounds, under reasonable conditions on the payoff matrices (discussed in Section 3), the play terminates in finite time with probability 1 due to collisions. We also do not assume any external model of punishment to prevent agents from deviating.

Finally, we note a superficial similarity of RUC games with stochastic games [32]. A stochastic game is a repeated game with an underlying state space. In each round, players simultaneously choose actions from some action set, based on which they get payoffs and the

state of the game changes stochastically according to some transition matrix. The utility to a player is typically assumed to be the discounted sum of the payoffs per round. Stochastic games are known to admit Nash equilibria [32]. A key point of difference between stochastic games and RUC games is that literature on stochastic games only considers stationary strategies, whereas strategies in our RUC games do not have this restriction.

2 Preliminaries

Notation. We introduce some relevant notation.

1. For any boolean proposition P, let $\mathbf{1}(P)$ be 1 if P is true and 0 otherwise.
2. For $n \in \mathbb{Z}_{\geq 0}$, let $[1..n] := \{1, 2, \ldots, n\}$.
3. For any matrix $A \in \mathbb{R}^{m \times n}$, let $A_{i,j}$ or $A[i,j]$ denote the entry in the ith row and j^{th} column. For any vector $\mathrm{v} \in \mathbb{R}^n$, let v_i denote the ith entry of the vector.
4. For $n \in \mathbb{Z}_{\geq 1}$, let $\Delta_n := \{\mathrm{x} \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^n \mathrm{x}_i = 1\}$.
5. For any vector $\mathrm{v} \in \mathbb{R}^n$, let $\text{support}(\mathrm{v}) := \{i \in [1..n] : \mathrm{v}_i \neq 0\}$. v is said to have *full support* if $\text{support}(\mathrm{v}) = [1..n]$.
6. For a vector $\mathrm{v} \in \mathbb{R}^n$, define $\|\mathrm{v}\|_1 := \sum_{i=1}^n |\mathrm{v}_i|$.
7. For any $i \in \mathbb{Z}_{\geq 1}$, let $\mathrm{e}^{(i)}$ be a vector such that $\mathrm{e}^{(i)}_i = 1$ and $\mathrm{e}^{(i)}_j = 0$ for all $j \neq i$.

Two-Player Games. We study two-player games between a *max player*, who is interested in maximizing her payoff, and a *min player*, who is interested in minimizing her payoff. We let $\mathcal{X}$ and $\mathcal{Y}$ denote the strategy space of the max and min players, respectively. When the max and min players use strategies $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, respectively, the max player gets a score $f_1(x, y)$ and the min player incurs a cost of $f_2(x, y)$, where these payoffs are given by functions $f_1, f_2 : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \cup \{\infty\}$. When $f_1 = f_2$, the game is said to be a *zero-sum* game.

Nash equilibrium. A Nash equilibrium (NE) is pair of strategies where no player can improve her payoff by unilaterally changing her strategy.

▶ **Definition 5.** *Let $f_1, f_2 : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \cup \{\infty\}$ be the players' payoff functions. The pair $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ is called a* Nash equilibrium *if no player can improve their payoff by switching to a different strategy. Formally,*

1. *(Max player cannot improve) $\forall x \in \mathcal{X}$, $f_1(x, y^*) \leq f_1(x^*, y^*)$.*
2. *(Min player cannot improve) $\forall y \in \mathcal{Y}$, $f_2(x^*, y) \geq f_2(x^*, y^*)$.*

While NE always exist for bimatrix games, they need not exist for general games. Moreover, NE may not be unique, i.e., multiple distinct NE may exist. We now define approximate NE.

▶ **Definition 6.** *Let $f_1, f_2 : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \cup \{\infty\}$ be the players' payoff functions. The pair $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ is called an ε-*approximate Nash equilibrium *if no player can improve their payoff by a factor of $(1 + \varepsilon)$ by switching to a different strategy. Formally,*

1. *(Max player cannot improve) $\forall x \in \mathcal{X}$, $f_1(x, y^*) \leq (1 + \varepsilon) \cdot f_1(x^*, y^*)$.*
2. *(Min player cannot improve) $\forall y \in \mathcal{Y}$, $f_2(x^*, y) \geq (1 + \varepsilon)^{-1} \cdot f_2(x^*, y^*)$.*

Zero-Sum Games. For a two-player zero-sum game, the following theorem shows that all Nash equilibria give the same payoffs to the agents, so the agents don't prefer any one of them over the other.

▶ **Proposition 7.** *Let (f, f) be a zero-sum game, where $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \cup \{\infty\}$. If (f, f) admits multiple Nash equilibria, they have the same payoff. Formally, if (x_1, y_1) and (x_2, y_2) are Nash equilibria, then $f(x_1, y_1) = f(x_2, y_2) = f(x_1, y_2) = f(x_2, y_1)$. Moreover, (x_1, y_2) and (x_2, y_1) are also Nash equilibria.*

Proof. Since (x_1, y_1) is a Nash equilibrium, the max player doesn't gain by switching to x_2 and min player doesn't gain by switching to y_2. Hence, $f(x_2, y_1) \leq f(x_1, y_1) \leq f(x_1, y_2)$. Since (x_2, y_2) is a Nash equilibrium, the max player doesn't gain by switching to x_1 and the min player doesn't gain by switching to y_1. Hence, $f(x_1, y_2) \leq f(x_2, y_2) \leq f(x_2, y_1)$. Combining these inequalities gives us $f(x_1, y_1) = f(x_2, y_2) = f(x_1, y_2) = f(x_2, y_1)$.

For any x and y, we get $f(x, y_2) \leq f(x_2, y_2) = f(x_1, y_2) = f(x_1, y_1) \leq f(x_1, y)$, and $f(x, y_1) \leq f(x_1, y_1) = f(x_2, y_1) = f(x_2, y_2) \leq f(x_2, y)$. Hence, (x_1, y_2) and (x_2, y_1) are also Nash equilibria. ◀

3 Nash Equilibria in Stationary RUC games

In this section, we study Nash equilibria for SRUC games, i.e., an RUC game (A, B) where agents are *forced* to use only stationary strategies. Let $e^{(A)}(\mathrm{x}, \mathrm{y})$ and $e^{(B)}(\mathrm{x}, \mathrm{y})$ be the max player's expected total score and the min player's expected total cost, when the max and min players play stationary strategies x and y, respectively.

▶ **Lemma 8.** *Let (A, B) represent an SRUC game. Let* x *and* y *be the max player's and min player's stationary strategies, respectively. Then for $C \in \{A, B\}$, we have*

$$e^{(C)}(\mathrm{x}, \mathrm{y}) = \begin{cases} \dfrac{\mathrm{x}^T C \mathrm{y}}{\mathrm{x}^T \mathrm{y}} & \text{if } \mathrm{x}^T\mathrm{y} > 0 \\ \infty & \text{if } \mathrm{x}^T\mathrm{y} = 0 \text{ and } \mathrm{x}^T C \mathrm{y} > 0 \\ 0 & \text{if } \mathrm{x}^T\mathrm{y} = \mathrm{x}^T C \mathrm{y} = 0 \end{cases}.$$

Proof sketch. (See Appendix A of the full version [23] for the full proof.)

The max player's expected per-round score is $\mathrm{x}^T A \mathrm{y}$, and the probability of collision in a round is $\mathrm{x}^T\mathrm{y}$. If $\mathrm{x}^T\mathrm{y} = 0$, a collision never happens, and so her total score is ∞ or 0 depending on whether $\mathrm{x}^T A \mathrm{y} > 0$. If $\mathrm{x}^T\mathrm{y} > 0$, then we can find $\mu := e^{(A)}(\mathrm{x}, \mathrm{y})$ by solving the equation $\mu = \mathrm{x}^T A \mathrm{y} + (1 - \mathrm{x}^T\mathrm{y})\mu$, which gives $\mu = \mathrm{x}^T A \mathrm{y} / \mathrm{x}^T\mathrm{y}$. The min player's expected total cost can be found analogously. ◀

From now on, instead of looking at SRUC games as multi-round games, we will treat them like single round games where the strategy space is Δ_n for both players, and the payoff functions are $e^{(A)}$ and $e^{(B)}$ for the max player and min player, respectively.

3.1 Existence of Nash Equilibrium

This section shows the existence of a Nash equilibrium in SRUC games. We first recall the definitions of an eigenvalue and eigenvector of a matrix.

▶ **Definition 9.** *For a square matrix $A \in \mathbb{R}^n$, complex number λ, and complex vector* $\mathrm{v} \in \mathbb{C}^n - \{0\}$, (λ, v) *is called an* eigenpair *of A (and λ is called an* eigenvalue *of A and* v *is called an* eigenvector *of A) if $A\mathrm{v} = \lambda v$.*

▶ **Lemma 10.** *For a square matrix A, λ is an eigenvalue of A iff λ is an eigenvalue of A^T.*

Proof. λ is an eigenvalue of A iff $\det(A - \lambda I) = 0$, and $\det(A - \lambda I) = \det(A^T - \lambda I)$. ◀

The following lemma establishes a sufficient condition for a strategy pair to be a Nash equilibrium of an SRUC game.

▶ **Lemma 11.** *Consider the SRUC game (A, B). Let (α, y^*) be an eigenpair for A such that $\|\mathrm{y}^*\|_1 = 1$ and $\mathrm{y}^*_i > 0$ for all $i \in [1..n]$. Let (β, x^*) be an eigenpair for B^T such that $\|\mathrm{x}^*\|_1 = 1$ and $\mathrm{x}^*_i > 0$ for all $i \in [1..n]$. Then $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium. Moreover, for any $\mathrm{x}, \mathrm{y} \in \Delta_n$, we have $e^{(A)}(\mathrm{x}, \mathrm{y}^*) = \alpha$ and $e^{(B)}(\mathrm{x}^*, \mathrm{y}) = \beta$.*

Proof. Let $\mathrm{x}, \mathrm{y} \in \Delta_n$. Since x^* and y^* have full support, ${\mathrm{x}^*}^T\mathrm{y} > 0$ and $\mathrm{x}^T\mathrm{y}^* > 0$. Also, by Lemma 8,

$$e^{(A)}(\mathrm{x}, \mathrm{y}^*) = \frac{\mathrm{x}^T A \mathrm{y}^*}{\mathrm{x}^T\mathrm{y}^*} = \frac{\mathrm{x}^T(\alpha \mathrm{y}^*)}{\mathrm{x}^T\mathrm{y}^*} = \alpha. \qquad e^{(B)}(\mathrm{x}^*, \mathrm{y}) = \frac{{\mathrm{x}^*}^T B \mathrm{y}}{{\mathrm{x}^*}^T\mathrm{y}} = \frac{(\beta \mathrm{x}^*)^T\mathrm{y}}{{\mathrm{x}^*}^T\mathrm{y}} = \beta.$$

Hence, $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium. ◀

We now show that such strategy pairs exist due to the Perron-Frobenius theorem stated below, thus proving the existence of Nash equilibrium for SRUC games in Theorem 14.

▶ **Theorem 12** (Perron-Frobenius [21])**.** *Let $A \in \mathbb{R}^{n \times n}_{\geq 0}$ be an irreducible matrix. Then*

1. *There exists a (unique) positive eigenvalue ρ of A, called the* Perron root *of A, such that for any other (possibly complex) eigenvalue λ of A, $|\lambda| \leq \rho$.*
2. *There exist unique vectors u and v such that $A^T\mathrm{u} = \rho\mathrm{u}$, $A\mathrm{v} = \rho\mathrm{v}$, and $\sum_{i=1}^n \mathrm{u}_i = \sum_{j=1}^n \mathrm{v}_j = 1$. u and v are called the left and right* Perron vectors *of A, respectively.*
3. *$\mathrm{u}_i > 0$ and $\mathrm{v}_i > 0$ for all $i \in [1..n]$.*
4. $\min_{i=1}^n \sum_{j=1}^n A_{i,j} \leq \rho \leq \max_{i=1}^n \sum_{j=1}^n A_{i,j}$, *and* $\min_{j=1}^n \sum_{i=1}^n A_{i,j} \leq \rho \leq \max_{j=1}^n \sum_{i=1}^n A_{i,j}$.

▶ **Definition 13.** *For irreducible matrices A and B, let* $\mathrm{perronSolution}(A, B)$ *be the tuple $(\rho_A, \rho_B, \mathrm{x}^*, \mathrm{y}^*)$, where ρ_A is the Perron root of A, y^* is the right Perron vector of A, ρ_B is the Perron root of B, and x^* is the left Perron vector of B.*

▶ **Theorem 14.** *For the SRUC game (A, B), where A and B are irreducible matrices, let $(\rho_A, \rho_B, \mathrm{x}^*, \mathrm{y}^*) = \mathrm{perronSolution}(A, B)$. Then $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium. Moreover, for any $\mathrm{x}, \mathrm{y} \in \Delta_n$, we have $e^{(A)}(\mathrm{x}, \mathrm{y}^*) = \rho_A$ and $e^{(B)}(\mathrm{x}^*, \mathrm{y}) = \rho_B$.*

Proof. Follows from Theorem 12 and Lemma 11. ◀

3.2 Uniqueness of Nash Equilibrium

In this section, we show that an SRUC game has a unique NE under mild assumptions on the payoff matrices. We begin with two lemmas which together show that any NE of an SRUC game has both players using strategies with full support, provided the payoff matrices A and B are irreducible and $\mathrm{graph}(A)$ is a subgraph of $\mathrm{graph}(B)$.

▶ **Lemma 15.** *Let $(\mathrm{x}^*, \mathrm{y}^*)$ be a Nash equilibrium for the SRUC game (A, B), where A is irreducible and* $\mathrm{graph}(A)$ *is a subgraph of* $\mathrm{graph}(B)$*. Then* $\mathrm{support}(\mathrm{y}^*) = [1..n]$.

Proof. The key idea is that if y^* doesn't have full support, then the max player is incentivized to play a strategy outside $\mathrm{support}(\mathrm{y}^*)$ to get an infinite score. Such a strategy would impose an infinite cost on the min player. Finally, it's possible for the min player to have a bounded cost by playing a full-support strategy.

For any $C \in \{A, B\}$, let $([1..n], E_C) := \text{graph}(C)$. Let $\overline{E}_C := [1..n] \times [1..n] \setminus E_C$. For any $\mathrm{x}, \mathrm{y} \in \Delta_n$ such that $\mathrm{x}^T\mathrm{y} = 0$, we get $\mathrm{x}^T C\mathrm{y} = 0 \iff \text{support}(\mathrm{x}) \times \text{support}(\mathrm{y}) \subseteq \overline{E}_C$. Since $E_A \subseteq E_B$, we get $\mathrm{x}^T A\mathrm{y} > 0 \implies \mathrm{x}^T B\mathrm{y} > 0$.

Let $S_y := \text{support}(\mathrm{y}^*)$. Suppose $S_y \neq [1..n]$. Then $\exists (i,j) \in E_A$ from $[1..n] \setminus S_y$ to S_y, since A is irreducible. Hence, $(\mathrm{e}^{(i)})^T A\mathrm{y}^* = \sum_{k \in S_y} A_{i,k}\mathrm{y}^*_k \geq A_{i,j}\mathrm{y}^*_j > 0$. Also, $(\mathrm{e}^{(i)})^T\mathrm{y}^* = 0$, since $i \notin S_y$. Hence, $e^{(A)}(\mathrm{e}^{(i)}, \mathrm{y}^*) = \infty$.

Since $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium, $\infty = e^{(A)}(\mathrm{e}^{(i)}, \mathrm{y}^*) \leq e^{(A)}(\mathrm{x}^*, \mathrm{y}^*)$. This is only possible if ${\mathrm{x}^*}^T\mathrm{y}^* = 0$ and ${\mathrm{x}^*}^T A\mathrm{y}^* > 0$. This means ${\mathrm{x}^*}^T B\mathrm{y}^* > 0$, and so $e^{(B)}(\mathrm{x}^*, \mathrm{y}^*) = \infty$.

Let $\widehat{\mathrm{y}}_j = 1/n$ for all $j \in [1..n]$. Then ${\mathrm{x}^*}^T\widehat{\mathrm{y}} > 0$, so $e^{(B)}(\mathrm{x}^*, \widehat{\mathrm{y}})$ is finite. Since $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium, we get $\infty = e^{(B)}(\mathrm{x}^*, \mathrm{y}^*) \leq e^{(B)}(\mathrm{x}^*, \widehat{\mathrm{y}}) \neq \infty$. This is a contradiction, so $\text{support}(\mathrm{y}^*) = [1..n]$. ◀

▶ **Lemma 16.** *Let* $(\mathrm{x}^*, \mathrm{y}^*)$ *be a Nash equilibrium for SRUC game* (A, B)*, where* B *is irreducible and* $\text{support}(\mathrm{y}^*) = [1..n]$*. Then* $\text{support}(\mathrm{x}^*) = [1..n]$*.*

Proof. The key idea is that if x^* doesn't have full support, then the min player can reduce her cost by not playing actions outside $\text{support}(\mathrm{x}^*)$.

Let $S_x := \text{support}(\mathrm{x}^*)$. Let $\alpha := \sum_{j \in S_x} \mathrm{y}^*_j$. Then $1 - \alpha = \sum_{j \in [1..n] \setminus S_x} \mathrm{y}^*_j$. Suppose $S_x \neq [1..n]$. Then $0 < \alpha < 1$, since $\text{support}(\mathrm{y}^*) = [1..n]$. Define vectors $\widehat{\mathrm{y}}$ and $\widetilde{\mathrm{y}}$ as follows:

$$\widehat{\mathrm{y}}_j = \begin{cases} \mathrm{y}^*_j/\alpha & \text{if } j \in S_x \\ 0 & \text{if } j \notin S_x \end{cases}, \qquad \widetilde{\mathrm{y}}_j = \begin{cases} 0 & \text{if } j \in S_x \\ \mathrm{y}^*_j/(1-\alpha) & \text{if } j \notin S_x \end{cases}.$$

Then $\mathrm{y}^* = \alpha\widehat{\mathrm{y}} + (1-\alpha)\widetilde{\mathrm{y}}$ and ${\mathrm{x}^*}^T\mathrm{y}^* = \alpha{\mathrm{x}^*}^T\widehat{\mathrm{y}}$. Hence,

$$e^{(B)}(\mathrm{x}^*, \mathrm{y}^*) = \frac{{\mathrm{x}^*}^T B\mathrm{y}^*}{{\mathrm{x}^*}^T\mathrm{y}^*} = \frac{\alpha{\mathrm{x}^*}^T B\widehat{\mathrm{y}} + (1-\alpha){\mathrm{x}^*}^T B\widetilde{\mathrm{y}}}{\alpha{\mathrm{x}^*}^T\widehat{\mathrm{y}}} = e^{(B)}(\mathrm{x}^*, \widehat{\mathrm{y}}) + \frac{1-\alpha}{\alpha}\frac{{\mathrm{x}^*}^T B\widetilde{\mathrm{y}}}{{\mathrm{x}^*}^T\widehat{\mathrm{y}}}.$$

Since $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium, $e^{(B)}(\mathrm{x}^*, \mathrm{y}^*) \leq e^{(B)}(\mathrm{x}^*, \widehat{\mathrm{y}})$. Hence, ${\mathrm{x}^*}^T B\widetilde{\mathrm{y}} = 0$. However,

$${\mathrm{x}^*}^T B\widetilde{\mathrm{y}} = \sum_{i \in S_x} \sum_{j \in [1..n] \setminus S_x} \mathrm{x}^*_i B_{i,j}\widetilde{\mathrm{y}}_j.$$

Since B is irreducible, there is an edge (i,j) in $\text{graph}(B)$ from S_x to $[1..n] \setminus S_x$. Therefore, ${\mathrm{x}^*}^T B\widetilde{\mathrm{y}} \geq B_{i,j}\mathrm{x}^*_i\widetilde{\mathrm{y}}_j > 0$, which is a contradiction. Hence, $\text{support}(\mathrm{x}^*) = [1..n]$. ◀

We now state a simple result about the ratio of sums (Lemma 17), and use it to prove that full support Nash equilibria must be eigenvectors (Lemma 18).

▶ **Lemma 17.** *Let* $\mathrm{u} \in \mathbb{R}^n$ *and* $\mathrm{v} \in \mathbb{R}^n_{>0}$*. Let*

$$\alpha := \min_{i=1}^{n} \frac{\mathrm{u}_i}{\mathrm{v}_i} \qquad \beta := \max_{i=1}^{n} \frac{\mathrm{u}_i}{\mathrm{v}_i} \qquad z := \frac{\sum_{i=1}^n \mathrm{u}_i}{\sum_{i=1}^n \mathrm{v}_i}.$$

Then either $z = \alpha = \beta$ *or* $\alpha < z < \beta$*.*

Proof. If $\alpha = \beta$, then $\mathrm{u}_i/\mathrm{v}_i = \alpha$ for all $i \in [1..n]$, and hence $z = \alpha$. Now let $\alpha < \beta$. Then $\alpha\mathrm{v}_i \leq \mathrm{u}_i \leq \beta\mathrm{v}_i$ for all $i \in [1..n]$. Pick p and q such that $\mathrm{u}_p/\mathrm{v}_p = \alpha$ and $\mathrm{u}_q/\mathrm{v}_q = \beta$. Then

$$\sum_{i=1}^n \mathrm{u}_i \leq \alpha\mathrm{v}_p + \sum_{i \in [1..n] \setminus \{p\}} \beta\mathrm{v}_i < \beta\sum_{i=1}^n \mathrm{v}_i, \qquad \sum_{i=1}^n \mathrm{u}_i \geq \sum_{i \in [1..n] \setminus \{q\}} \alpha\mathrm{v}_i + \beta\mathrm{v}_q > \alpha\sum_{i=1}^n \mathrm{v}_i.$$

Hence, $\alpha < z < \beta$. ◀

▶ **Lemma 18.** *Let* $(\mathrm{x}^*, \mathrm{y}^*)$ *be a Nash equilibrium for SRUC game* (A, B) *such that* $\mathrm{support}(\mathrm{x}^*) = \mathrm{support}(\mathrm{y}^*) = [1..n]$. *Then* $(e^{(B)}(\mathrm{x}^*, \mathrm{y}^*), \mathrm{x}^*)$ *is an eigenpair of* B^T *and* $(e^{(A)}(\mathrm{x}^*, \mathrm{y}^*), \mathrm{y}^*)$ *is an eigenpair of* A.

Proof. For $i \in [1..n]$, let $\mathrm{w}_i := \mathrm{x}^*_i \mathrm{y}^*_i$. Then $\mathrm{w}_i > 0$ for all $i \in [1..n]$.

For $j \in [1..n]$, let $\mathrm{u}_j := (B^T\mathrm{x}^*)_j \mathrm{y}^*_j$. Let

$$\alpha_x := \min_{j=1}^{n} \frac{\mathrm{u}_j}{\mathrm{w}_j} = \min_{j=1}^{n} \frac{(B^T\mathrm{x}^*)_j}{\mathrm{x}^*_j} \qquad \beta_x := \max_{j=1}^{n} \frac{\mathrm{u}_j}{\mathrm{w}_j} = \max_{j=1}^{n} \frac{(B^T\mathrm{x}^*)_j}{\mathrm{x}^*_j}$$

Let p and q be indices such that $\mathrm{u}_p/\mathrm{w}_p = \alpha_x$ and $\mathrm{u}_q/\mathrm{w}_q = \beta_x$. Then $e^{(B)}(\mathrm{x}^*, \mathrm{e}^{(p)}) = \alpha_x$, $e^{(B)}(\mathrm{x}^*, \mathrm{e}^{(q)}) = \beta_x$, and $e^{(B)}(\mathrm{x}^*, \mathrm{y}^*) = (\sum_{j=1}^{n} \mathrm{u}_j)/(\sum_{j=1}^{n} \mathrm{w}_j)$. Suppose $\alpha_x \neq \beta_x$. Then by Lemma 17, $e^{(B)}(\mathrm{x}^*, \mathrm{e}^{(p)}) < e^{(B)}(\mathrm{x}^*, \mathrm{y}^*) < e^{(B)}(\mathrm{x}^*, \mathrm{e}^{(q)})$. This contradicts the fact that $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium. Hence, $\alpha_x = \beta_x = e^{(B)}(\mathrm{x}^*, \mathrm{y}^*)$ and $(B^T\mathrm{x}^*)_j/\mathrm{x}^*_j = \alpha_x$ for all $j \in [1..n]$. Hence, (α_x, x^*) is an eigenpair of B^T.

For $i \in [1..n]$, let $\mathrm{v}_i := \mathrm{x}^*_i (A\mathrm{y}^*)_i$. Let

$$\alpha_y := \min_{i=1}^{n} \frac{\mathrm{v}_i}{\mathrm{w}_i} = \min_{i=1}^{n} \frac{(A\mathrm{y}^*)_i}{\mathrm{y}^*_i} \qquad \beta_y := \max_{i=1}^{n} \frac{\mathrm{v}_i}{\mathrm{w}_i} = \max_{i=1}^{n} \frac{(A\mathrm{y}^*)_i}{\mathrm{y}^*_i}$$

We can similarly show that $\alpha_y = \beta_y = e^{(A)}(\mathrm{x}^*, \mathrm{y}^*)$ and (α_y, y^*) is an eigenpair of A. ◀

Finally, we show that non-negative eigenvectors are essentially Perron vectors (Lemma 19), and use this result to establish uniqueness of Nash equilibrium (Theorem 20).

▶ **Lemma 19.** *Let* $A \in \mathbb{R}^{n \times n}_{\geq 0}$ *be an irreducible matrix. Let* ρ *be the Perron root of* A*, and* u *and* v *be the left and right Perron vectors of* A*, respectively. Then both of these hold:*

1. *If* (λ, y) *is an eigenpair of* A *such that* $\mathrm{y}_i \geq 0\ \forall i$ *and* $\|\mathrm{y}\|_1 = 1$*, then* $\lambda = \rho$ *and* $\mathrm{y} = \mathrm{v}$.
2. *If* (λ, x) *is an eigenpair of* A^T *such that* $\mathrm{x}_i \geq 0\ \forall i$ *and* $\|\mathrm{x}\|_1 = 1$*, then* $\lambda = \rho$ *and* $\mathrm{x} = \mathrm{u}$.

Proof. $\rho \mathrm{u}^T\mathrm{y} = (A^T\mathrm{u})^T\mathrm{y} = \mathrm{u}^T(A\mathrm{y}) = \lambda \mathrm{u}^T\mathrm{y}$. Since $\mathrm{support}(\mathrm{u}) = [1..n]$, we get $\mathrm{u}^T\mathrm{y} > 0$. Hence, $\lambda = \rho$. By Theorem 12, v is the unique eigenvector corresponding to eigenvalue ρ such that $\|\mathrm{v}\|_1 = 1$. Hence, $\mathrm{y} = \mathrm{v}$. We get part 2 of the lemma by applying part 1 on A^T. ◀

▶ **Theorem 20.** *Let* (A, B) *be an SRUC game, where* A *and* B *are irreducible and* $\mathrm{graph}(A)$ *is a subgraph of* $\mathrm{graph}(B)$*. Let* $(\rho_A, \rho_B, \mathrm{x}^*, \mathrm{y}^*) = \mathrm{perronSolution}(A, B)$*. Then* $(\mathrm{x}^*, \mathrm{y}^*)$ *is the unique Nash equilibrium for this SRUC game.*

Proof. Let $(\widehat{\mathrm{x}}, \widehat{\mathrm{y}})$ be any Nash equilibrium for SRUC game (A, B). From Lemmas 15 and 16, we get $\mathrm{support}(\widehat{\mathrm{x}}) = \mathrm{support}(\widehat{\mathrm{y}}) = [1..n]$.

Let $\sigma_C := e^{(C)}(\widehat{\mathrm{x}}, \widehat{\mathrm{y}})$ for $C \in \{A, B\}$. By Lemma 18, $(\sigma_B, \widehat{\mathrm{x}})$ is an eigenpair of B^T and $(\sigma_A, \widehat{\mathrm{y}})$ is an eigenpair of A. By Lemma 19, we get $\sigma_B = \rho_B$, $\widehat{\mathrm{x}} = \mathrm{x}^*$, $\sigma_A = \rho_A$, $\widehat{\mathrm{y}} = \mathrm{y}^*$. ◀

Theorem 20 shows uniqueness of NE of an SRUC game (A, B) under two conditions: (i) A and B are irreducible, and (ii) $\mathrm{graph}(A) \subseteq \mathrm{graph}(B)$. The following two lemmas show that if either of these conditions is relaxed, we can no longer guarantee uniqueness of NE.

▶ **Lemma 21.** *Let* (A, B) *be an SRUC game. Suppose* $\exists (i, j)$ *such that* $i \neq j$*,* $A_{i,j} > 0$*, and* $B_{i,j} = 0$*. Then* $(\mathrm{e}^{(i)}, \mathrm{e}^{(j)})$ *is a Nash equilibrium. Moreover,* $e^{(A)}(\mathrm{e}^{(i)}, \mathrm{e}^{(j)}) = \infty$ *and* $e^{(B)}(\mathrm{e}^{(i)}, \mathrm{e}^{(j)}) = 0$.

Proof. ${\mathrm{e}^{(i)}}^T A \mathrm{e}^{(j)} = A[i, j] > 0$, ${\mathrm{e}^{(i)}}^T B \mathrm{e}^{(j)} = B[i, j] = 0$, and ${\mathrm{e}^{(i)}}^T \mathrm{e}^{(j)} = 0$. ◀

The NE in Lemma 21 is deterministic but the NE in Theorem 14 is randomized, implying that the game has at least two NE. Hence, the condition $\text{graph}(A) \subseteq \text{graph}(B)$ is necessary for uniqueness of NE if A and B are irreducible. The next lemma shows that if either A or B is reducible, then uniqueness of NE is not guaranteed.

▶ **Lemma 22.** *Let (A, A) be an SRUC game where $A := \left(\begin{smallmatrix} 1 & 0 \\ 0 & 2 \end{smallmatrix}\right)$. Then (x, y) is a Nash equilibrium iff* $\text{support}(\mathrm{x}) = \{1, 2\}$ *and* $\mathrm{y} = [1, 0]$.

Proof. Observe that A is reducible. For any $\mathrm{x}, \mathrm{y} \in \Delta_2$, if $\mathrm{x}_2 > 0$ and $\mathrm{y}_2 > 0$, then $e^{(A)}(\mathrm{x}, \mathrm{y}) = 1 + \mathrm{x}_2\mathrm{y}_2/(\mathrm{x}_1\mathrm{y}_1 + \mathrm{x}_2\mathrm{y}_2) > 1$. When $\mathrm{x}^T\mathrm{y} = 0$, we get $e^{(A)}(\mathrm{x}, \mathrm{y}) = 0$. Otherwise, $e^{(A)}(\mathrm{x}, \mathrm{y}) = 1$. This is summarized by the following table:

$x \setminus y$	$[1, 0]$	mix	$[0, 1]$
$[1, 0]$	1	1	0
mix	1	> 1	> 1
$[0, 1]$	0	> 1	> 1

This table shows that (x, y) is a NE iff $\text{support}(\mathrm{x}) = \{1, 2\}$ and $\mathrm{y} = [1, 0]$. ◀

The above example also shows that unlike the guarantee of Theorem 14, Nash equilibrium strategies may not have full support.

3.3 Approximate Nash Equilibrium

It may not be possible to compute Nash equilibria for SRUC games exactly, e.g., if eigenvectors are irrational. Hence, we would like to compute approximate Nash equilibria. We will use the following lemma.

▶ **Lemma 23.** *Let $\mathrm{x}^*, \widehat{\mathrm{x}} \in \Delta_n$ and $\delta \in [0, 1)$ such that* $\text{support}(\mathrm{x}^*) = [1..n]$ *and* $|\widehat{\mathrm{x}}_i - \mathrm{x}^*_i| \leq \delta \mathrm{x}^*_i$ *for all $i \in [1..n]$. Then* $\text{support}(\widehat{\mathrm{x}}) = [1..n]$ *and for any matrix $C \in \mathbb{R}^{n \times n}_{\geq 0}$ and any $j \in [1..n]$,*

$$\frac{(C^T\widehat{\mathrm{x}})_j}{\widehat{\mathrm{x}}_j} \in \left[\frac{1-\delta}{1+\delta}\frac{(C^T\mathrm{x}^*)_j}{\mathrm{x}^*_j}, \frac{1+\delta}{1-\delta}\frac{(C^T\mathrm{x}^*)_j}{\mathrm{x}^*_j}\right].$$

Proof. For any $i \in [1..n]$, we have $\widehat{\mathrm{x}}_i \in [(1-\delta)\mathrm{x}^*_i, (1+\delta)\mathrm{x}^*_i]$. Hence, $\text{support}(\widehat{\mathrm{x}}) = [1..n]$.

$$\begin{aligned}
&|(C^T\widehat{\mathrm{x}})_j - (C^T\mathrm{x}^*)_j| \leq |(C^T(\widehat{\mathrm{x}} - \mathrm{x}^*))_j| \leq \left|\sum_{i=1}^n C[i,j](\widehat{\mathrm{x}}_i - \mathrm{x}^*_i)\right| \\
&\leq \sum_{i=1}^n C[i,j]|\widehat{\mathrm{x}}_i - \mathrm{x}^*_i| \leq \delta \sum_{i=1}^n C[i,j]\mathrm{x}^*_i = \delta(C^T\mathrm{x}^*)_j. \\
&\implies (C^T\widehat{\mathrm{x}})_j \in [(1-\delta)(C^T\mathrm{x}^*)_j, (1+\delta)(C^T\mathrm{x}^*)_j] \\
&\implies \frac{(C^T\widehat{\mathrm{x}})_j}{\widehat{\mathrm{x}}_j} \in \left[\frac{1-\delta}{1+\delta}\frac{(C^T\mathrm{x}^*)_j}{\mathrm{x}^*_j}, \frac{1+\delta}{1-\delta}\frac{(C^T\mathrm{x}^*)_j}{\mathrm{x}^*_j}\right].
\end{aligned}$$ ◀

We now show that a close-enough approximation to a NE is an approximate NE of an SRUC game. This result allows us to compute approximate NE by using methods for approximately estimating the leading eigenvector of a matrix, such as power iteration [22].

▶ **Theorem 24.** *Let $(\mathrm{x}^*, \mathrm{y}^*)$ be a Nash equilibrium for the SRUC game (A, B) such that* $\text{support}(\mathrm{x}^*) = \text{support}(\mathrm{y}^*) = [1..n]$. *For $\delta \in [0, 1)$, let $\widehat{\mathrm{x}}, \widehat{\mathrm{y}} \in \Delta_n$ be such that for all $i \in [1..n]$, $|\widehat{\mathrm{x}}_i - \mathrm{x}^*_i| \leq \delta \mathrm{x}^*_i$ and $|\widehat{\mathrm{y}}_i - \mathrm{y}^*_i| \leq \delta \mathrm{y}^*_i$. Then $(\widehat{\mathrm{x}}, \widehat{\mathrm{y}})$ is a $\frac{4\delta}{(1-\delta)^2}$-approximate Nash equilibrium.*

Proof. By Lemma 18, (ρ_B, x^*) is an eigenpair of B^T. Hence, by Lemma 23, for all $j \in [1..n]$,

$$\frac{(B^T\widehat{\mathrm{x}})_j}{\widehat{\mathrm{x}}_j} \in \left[\frac{1-\delta}{1+\delta}\rho_B, \frac{1+\delta}{1-\delta}\rho_B\right].$$

For any $\mathrm{y} \in \Delta_n$, using Lemma 17, we get

$$e^{(B)}(\widehat{\mathrm{x}}, \mathrm{y}) = \frac{\sum_{j=1}^n (B^T\widehat{\mathrm{x}})_j \mathrm{y}_j}{\sum_{j=1}^n \widehat{\mathrm{x}}_j \mathrm{y}_j} \in \left[\min_{j=1}^n \frac{(B^T\widehat{\mathrm{x}})_j}{\widehat{\mathrm{x}}_j}, \max_{j=1}^n \frac{(B^T\widehat{\mathrm{x}})_j}{\widehat{\mathrm{x}}_j}\right] = \left[\frac{1-\delta}{1+\delta}\rho_B, \frac{1+\delta}{1-\delta}\rho_B\right].$$

In particular, for $\mathrm{y} = \widehat{\mathrm{y}}$, we get $e^{(B)}(\widehat{\mathrm{x}}, \widehat{\mathrm{y}}) \leq \frac{1+\delta}{1-\delta}\rho_B$. Let $\varepsilon := \frac{4\delta}{(1-\delta)^2}$. Thus, for any $\mathrm{y} \in \Delta_n$,

$$e^{(B)}(\widehat{\mathrm{x}}, \mathrm{y}) \geq \frac{1-\delta}{1+\delta}\rho_B \geq \left(\frac{1-\delta}{1+\delta}\right)^2 e^{(B)}(\widehat{\mathrm{x}}, \widehat{\mathrm{y}}) = \frac{e^{(B)}(\widehat{\mathrm{x}}, \widehat{\mathrm{y}})}{1+\varepsilon}.$$

We can analogously show that for any $\mathrm{x} \in \Delta_n$,

$$e^{(A)}(\mathrm{x}, \widehat{\mathrm{y}}) \leq \left(\frac{1+\delta}{1-\delta}\right)^2 e^{(A)}(\widehat{\mathrm{x}}, \widehat{\mathrm{y}}) = (1+\varepsilon) \cdot e^{(A)}(\widehat{\mathrm{x}}, \widehat{\mathrm{y}}).$$

Hence, $(\widehat{\mathrm{x}}, \widehat{\mathrm{y}})$ is an ε-approximate NE. ◀

4 Nash Equilibria in General RUC games

In this section, we study RUC games when players are allowed to use non-stationary strategies. As discussed in the Introduction, the strategy space can be complicated in general RUC games. We first describe a framework for defining strategies in a RUC game.

4.1 Strategy Space of RUC games

A deterministic (possibly non-stationary) strategy in an RUC game is a function f that takes as input a list of actions played by the opponent so far and outputs the next action for the player. E.g., if the max player is using a deterministic strategy f, then in the kth round, the max player will play the action $f(J)$, where $J := [j_1, j_2, \ldots, j_{k-1}]$ is the list of actions played by the min player in the first $k-1$ rounds.

A randomized strategy is a distribution over deterministic strategies. Formally, let $[1..n]^*$ be the set of all finite lists where each element is in $[1..n]$. Let Ω be the set of all deterministic strategies, i.e., the set of all functions from $[1..n]^*$ to $[1..n]$. Then a randomized strategy is given by a probability space $(\Omega, \mathcal{E}, P)$. Recall that in a probability space, $\mathcal{E} \subseteq 2^\Omega$ is the set of events and $P : \mathcal{E} \to [0,1]$ is a probability measure. If a randomized strategy f is sampled from this probability space, then for any set $F \in \mathcal{E}$ of deterministic strategies, we say $\Pr(f \in F) := P(F)$.

For any $\mathrm{x} \in \Delta_n$, and any random variable $X \in [1..n]$, we write $X \sim \mathrm{x}$ to say that $\Pr(X = i) = \mathrm{x}_i$ for all $i \in [1..n]$. The length of a list L, denoted by $|L|$, is the number of elements in L. Unless specified otherwise, assume all lists are finite.

▶ **Definition 25** (stationary strategy). *Let $\mathrm{x} \in \Delta_n$. Let $[I_0, I_1, \ldots]$ be an infinite sequence of independent random variables where for each $t \in \mathbb{Z}_{\geq 0}$, $I_t \in [1..n]$ and $I_t \sim \mathrm{x}$. Then* $\mathrm{stat}(\mathrm{x})$ *is a strategy f where $f(J) := I_{|J|}$.* $\mathrm{stat}(\mathrm{x})$ *is called the* stationary strategy *for parameter* x.

With a little abuse of notation, we will sometimes write x instead of $\mathrm{stat}(\mathrm{x})$.

Residual Strategies. For any two lists L_1 and L_2, let $L_1 + L_2$ denote their concatenation. Let $\emptyset$ denote the empty list. Let $[x]$ denote a list of length 1 containing the element x.

Given a list $K \in [1..n]^*$ and a strategy f, define the function f^K as $f^K(K') := f(K+K')$. Intuitively, if a player is using strategy f, then after the opponent has played actions K, f^K is the strategy for the remaining game. f^K is called the K-*residual* strategy of f. Due to the recursive nature of RUC games, residual strategies are helpful in their analysis.

Let $I := [i_1, \ldots, i_k]$ and $J := [j_1, \ldots, j_k]$ be lists. Let f be a randomized strategy. We want to define $f^{(I,J)}$ as the strategy f^J conditioned on the player responding with actions I when the opponent plays actions J. Formally, let $\text{isResp}(f, I, J)$ be the event that a player using strategy f responds with actions I to opponent's actions J, i.e., $\forall t \in [1..k], f([j_1, \ldots, j_{t-1}]) = i_t$. Call (I, J) a *feasible history* for f if $\Pr(\text{isResp}(f, I, J)) > 0$. For any feasible history (I, J) of f, define $f^{(I,J)}$, called the (I, J)-*residual strategy* of f, as a strategy having distribution

$$\Pr(f^{(I,J)} \in F) := \Pr(f^J \in F \mid \text{isResp}(f, I, J)).$$

Intuitively, stationary strategies should remain unchanged when conditioned on past actions. We prove this formally.

▶ **Lemma 26.** $\text{stat(x)}^{(I,J)}$ *has the same distribution as* stat(x).

Proof. (See Appendix C.1 of the full version [23].) ◀

Expected Score. Consider the RUC game (A, B), where A and B are $n \times n$ matrices. Assume that the max player's and min player's strategies are independent. We now formally define the *expected score (or cost) of a pair of strategies*, and obtain a recursive expression for it to make analysis easier.

▶ **Definition 27** (score). *Let $S^{(A,r)}(f, g)$ and $S^{(B,r)}(f, g)$ be the max player's total score and min player's total cost, respectively, in the first r rounds of the RUC game (A, B) when the max player uses strategy f and the min player uses strategy g. (Note that when f and g are randomized strategies, $S^{(A,r)}(f, g)$ and $S^{(B,r)}(f, g)$ are random variables.) For $C \in \{A, B\}$, let $e^{(C,r)}(f, g) := \mathbb{E}(S^{(C,r)}(f, g))$ and $e^{(C,\infty)}(f, g) := \lim_{r\to\infty} e^{(C,r)}(f, g)$.*

▶ **Observation 28.** *The sequence $[S^{(C,r)}(f, g)]_{r=0}^{\infty}$ is monotonically increasing, so the sequence $[e^{(C,r)}(f, g)]_{r=0}^{\infty}$ is also monotonically increasing. By the monotone convergence theorem, $e^{(C,\infty)}(f, g) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$, i.e., the sequence $[e^{(C,r)}(f, g)]_{r=0}^{\infty}$ either has a non-negative limit or is unbounded.*

▶ **Observation 29.** *Let f and g be strategies of the max and min players, respectively. Let $u := f(\emptyset)$ and $v := g(\emptyset)$. Then for $C \in \{A, B\}$,*

$$S^{(C,r)}(f, g) = \begin{cases} C[u, v] + \mathbf{1}(u \neq v) S^{(C,r-1)}(f^{[v]}, g^{[u]}) & \text{if } r > 0 \\ 0 & \text{otherwise} \end{cases}.$$

▶ **Lemma 30.** *Let f and g be independent strategies of the max and min players, respectively. Let $f(\emptyset) \sim \text{x}$, $g(\emptyset) \sim \text{y}$, $S_x := \text{support(x)}$, and $S_y := \text{support(y)}$. Then for $C \in \{A, B\}$,*

$$e^{(C,r)}(f, g) = \begin{cases} \text{x}^T C \text{y} + \sum_{i \in S_x} \sum_{j \in S_y} \text{x}_i \text{y}_j \mathbf{1}(i \neq j) e^{(C,r-1)}(f^{([i],[j])}, g^{([j],[i])}) & \text{if } r > 0 \\ 0 & \text{otherwise} \end{cases},$$

$$e^{(C,\infty)}(f, g) = \text{x}^T C \text{y} + \sum_{i \in S_x} \sum_{j \in S_y} \text{x}_i \text{y}_j \mathbf{1}(i \neq j) e^{(C,\infty)}(f^{([i],[j])}, g^{([j],[i])}).$$

Proof sketch. Follows from Observation 29. See Appendix C.1 of the full version [23] for the full proof. ◀

4.2 Existence of Nash Equilibrium

To see whether a pair of stationary strategies can give us a Nash equilibrium, we first investigate (in the next two lemmas) the upper and lower bounds on a player's payoff when she is free to play any strategy (even non-stationary ones) and her opponent uses a stationary strategy with full support.

▶ **Lemma 31.** *Let (A, B) be an RUC game. Let $\widehat{\mathrm{x}} \in \Delta_n$ such that* $\mathrm{support}(\widehat{\mathrm{x}}) = [1..n]$. *Let $\alpha := \min_{i=1}^n (B^T\widehat{\mathrm{x}})_i/\widehat{\mathrm{x}}_i$ and $\beta := \max_{i=1}^n (B^T\widehat{\mathrm{x}})_i/\widehat{\mathrm{x}}_i$. Then for any strategy g, we have $\alpha \le e^{(B,\infty)}(\widehat{\mathrm{x}}, g) \le \beta$.*

Proof. By the definition of α and β, and by Lemma 17, we get that for any $\mathrm{y} \in \Delta_n$, we have $\alpha\widehat{\mathrm{x}}^T\mathrm{y} \le \widehat{\mathrm{x}}^T B\mathrm{y} \le \beta\widehat{\mathrm{x}}^T\mathrm{y}$.

Define the predicate $P(r) : \forall g, e^{(B,r)}(\widehat{\mathrm{x}}, g) \le \beta$. We will prove $P(r)$ by induction, and that would imply $P(\infty)$. The base case holds, since $e^{(B,0)}(\widehat{\mathrm{x}}, g) = 0 \le \beta$. Now fix $r \ge 1$, g, and let $g(\emptyset) \sim \mathrm{y}$ and $S_y := \mathrm{support}(\mathrm{y})$. Then by Lemma 30,

$$\begin{aligned} e^{(B,r)}(\widehat{\mathrm{x}}, g) &= \widehat{\mathrm{x}}^T B\mathrm{y} + \sum_{i=1}^n \sum_{j \in S_y} \widehat{\mathrm{x}}_i \mathrm{y}_j \mathbf{1}(i \ne j) e^{(B,r-1)}(\widehat{\mathrm{x}}, g^{([j],[i])}) \\ &\le \beta\widehat{\mathrm{x}}^T\mathrm{y} + \sum_{i=1}^n \sum_{j=1}^n \widehat{\mathrm{x}}_i \mathrm{y}_j \mathbf{1}(i \ne j)\beta \ = \beta. \qquad \text{(by inductive hypothesis)} \end{aligned}$$

This proves $P(r)$, and hence, also proves $P(\infty)$.

Next, we need to prove that $e^{(B,\infty)}(\widehat{\mathrm{x}}, g) \ge \alpha$ for all g. Let $\mu := \inf_g e^{(B,\infty)}(\widehat{\mathrm{x}}, g)$. Pick any $\varepsilon > 0$. Let $\widehat{g}$ be a strategy such that $e^{(B,\infty)}(\widehat{\mathrm{x}}, \widehat{g}) \le \mu + \varepsilon$. Let $\widehat{g}(\emptyset) \sim \widehat{\mathrm{y}}$ and $S_y := \mathrm{support}(\widehat{\mathrm{y}})$. Let $\gamma := \min_{i=1}^n \widehat{\mathrm{x}}_i$. Then $\widehat{\mathrm{x}}^T\widehat{\mathrm{y}} \ge \gamma$, and by Lemma 30,

$$\begin{aligned} \mu + \varepsilon \ge e^{(B,\infty)}(\widehat{\mathrm{x}}, \widehat{g}) &= \widehat{\mathrm{x}}^T B\widehat{\mathrm{y}} + \sum_{i=1}^n \sum_{j \in S_y} \widehat{\mathrm{x}}_i \widehat{\mathrm{y}}_j \mathbf{1}(i \ne j) e^{(B,\infty)}(\widehat{\mathrm{x}}, \widehat{g}^{([j],[i])}) \\ &\ge \alpha\widehat{\mathrm{x}}^T\widehat{\mathrm{y}} + \sum_{i=1}^n \sum_{j=1}^n \widehat{\mathrm{x}}_i \widehat{\mathrm{y}}_j \mathbf{1}(i \ne j)\mu \ = \mu + \widehat{\mathrm{x}}^T\widehat{\mathrm{y}}(\alpha - \mu). \end{aligned}$$

Since $\mathrm{support}(\widehat{\mathrm{x}}) = [1..n]$, we get $\gamma > 0$ and $\alpha - \mu \le \varepsilon/\widehat{\mathrm{x}}^T\widehat{\mathrm{y}} \le \varepsilon/\gamma$. Since this is true for all $\varepsilon > 0$, we get $\alpha \le \mu$. Hence, $\alpha \le e^{(B,\infty)}(\widehat{\mathrm{x}}, g) \le \beta$ for every strategy g. ◀

▶ **Lemma 32.** *Let (A, B) be an RUC game. Let $\widehat{\mathrm{y}} \in \Delta_n$ such that* $\mathrm{support}(\widehat{\mathrm{y}}) = [1..n]$. *Let $\alpha := \min_{i=1}^n (A\widehat{\mathrm{y}})_i/\widehat{\mathrm{y}}_i$ and $\beta := \max_{i=1}^n (A\widehat{\mathrm{y}})_i/\widehat{\mathrm{y}}_i$. Then for any strategy f, we have $\alpha \le e^{(A,\infty)}(f, \widehat{\mathrm{y}}) \le \beta$.*

Proof. (Similar to the proof of Lemma 31.) ◀

Next, we identify sufficient conditions (as in Lemma 11 in Section 3) to make the upper and lower bounds in Lemmas 31 and 32 coincide, which gives us a Nash equilibrium.

▶ **Lemma 33.** *Consider the RUC game (A, B). Let (α, y^*) be an eigenpair for A such that $\|\mathrm{y}^*\|_1 = 1$ and $\mathrm{y}_i^* > 0$ for all $i \in [1..n]$. Let (β, x^*) be an eigenpair for B^T such that $\|\mathrm{x}^*\|_1 = 1$ and $\mathrm{x}_i^* > 0$ for all $i \in [1..n]$. Then $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium. Moreover, for any strategies f and g, we have $e^{(A,\infty)}(f, \mathrm{y}^*) = \alpha$ and $e^{(B,\infty)}(\mathrm{x}^*, g) = \beta$.*

Proof. Follows from Lemmas 31 and 32. ◀

Next, we show that conditions of Lemma 33 can be satisfied using the Perron-Frobenius theorem (Theorem 12), so a Nash equilibrium given by stationary strategies always exists.

▶ **Theorem 34.** *Let (A, B) be an RUC game where A and B are irreducible (c.f. Definition 1). Let $(\rho_A, \rho_B, \mathrm{x}^*, \mathrm{y}^*) = \mathrm{perronSolution}(A, B)$. Then $(\mathrm{x}^*, \mathrm{y}^*)$ is a Nash equilibrium. Moreover, for any strategies f and g, we have $e^{(A,\infty)}(f, \mathrm{y}^*) = \rho_A$ and $e^{(B,\infty)}(\mathrm{x}^*, g) = \rho_B$.*

Proof. Follows from Lemma 33. ◀

Note that for any stationary strategies x and y, if (x, y) is a Nash equilibrium for the RUC game (A, B), then (x, y) is also a Nash equilibrium for the SRUC game (A, B). This is because if a player cannot improve her payoff by switching to a different strategy, then she also cannot improve her payoff by switching to a different stationary strategy. Hence, Theorem 14 is a corollary of Theorem 34.

Since computing Nash equilibrium exactly may be hard, we consider approximate Nash equilibria. To do this, we generalize the corresponding result about SRUC games (Theorem 24 in Section 3) to RUC games.

▶ **Theorem 35** (approximate Nash equilibrium)**.** *Let $(\mathrm{stat}(\mathrm{x}^*), \mathrm{stat}(\mathrm{y}^*))$ be a Nash equilibrium for the RUC game (A, B) such that $\mathrm{support}(\mathrm{x}^*) = \mathrm{support}(\mathrm{y}^*) = [1..n]$. For $\delta \in [0, 1)$, let $\widehat{\mathrm{x}}, \widehat{\mathrm{y}} \in \Delta_n$ such that for all $i \in [1..n]$, $|\widehat{\mathrm{x}}_i - \mathrm{x}^*_i| \le \delta \mathrm{x}^*_i$ and $|\widehat{\mathrm{y}}_i - \mathrm{y}^*_i| \le \delta \mathrm{y}^*_i$. Then $(\widehat{\mathrm{x}}, \widehat{\mathrm{y}})$ is a $\frac{4\delta}{(1-\delta)^2}$-approximate Nash equilibrium for the RUC game (A, B).*

Proof sketch. The proof is similar to Theorem 24. It follows from Lemmas 18, 23, 31, and 32. See Appendix C.2 of the full version [23] for the full proof. ◀

4.3 Uniqueness of Nash Equilibrium up to Equivalence

Before we investigate the uniqueness of Nash equilibrium for RUC games, we first describe a phenomenon where two different randomized strategies can behave similarly.

▶ **Definition 36** (collisions)**.** *For lists $I := [i_1, \ldots, i_k]$ and $J := [j_1, \ldots, j_k]$, let $\mathrm{collisions}(I, J)$ be the number of collisions if the max player and min player play actions I and J, respectively. Formally, $\mathrm{collisions}(I, J) := \sum_{t=1}^{k} \mathbf{1}(i_t = j_t)$.*

The pair (I, J) is called collision-consistent *if $i_t = j_t$ for all $t < k$.*

▶ **Definition 37.** *Two randomized strategies f_1 and f_2 are said to be* equivalent *if for every collision-consistent pair (I, J), we have $\Pr(\mathrm{isResp}(f_1, I, J)) = \Pr(\mathrm{isResp}(f_2, I, J))$.*

▶ **Example 38.** Let $n = 3$. For any $u, v \in \{1, 2\}$, let $f_{u,v}$ be the deterministic strategy where $f_{u,v}(J) = u$ if $J = [2]$, $f_{u,v}(J) = v$ if $J = [3]$, and $f_{u,v}(J) = 1$ otherwise.

For any $p \in [0, 1/2]$, let h_p be a randomized strategy where $\Pr(h_p = f_{1,1}) = \Pr(h_p = f_{2,2}) = p$ and $\Pr(h_p = f_{1,2}) = \Pr(h_p = f_{2,1}) = 1/2 - p$. Then for any lists I and J of the same length, $\Pr(\mathrm{isResp}(h_p, I, J))$ is not a function of p. Hence, h_0 and $h_{1/2}$ are equivalent.

We now show that given a pair of strategies, replacing each strategy by an equivalent strategy makes no difference to anyone's payoff.

▶ **Lemma 39.** *Let f_1 and f_2 be equivalent strategies of the max player and g_1 and g_2 be equivalent strategies of the min player. Then for the RUC game (A, B), we have $e^{(C,r)}(f_1, g_1) = e^{(C,r)}(f_2, g_2)$ for all $C \in \{A, B\}$ and $r \in \mathbb{Z}_{\ge 0} \cup \{\infty\}$.*

Proof sketch. For any $r \in \mathbb{Z}_{\geq 0}$, the expected score in the first r rounds of a pair of strategies only depends on the distribution of collision-consistent pairs (I, J) where $|I| = |J| \leq r$. Hence, $e^{(C,r)}(f_1, g_1) = e^{(C,r)}(f_2, g_2)$. Take the limit $r \to \infty$ to get $e^{(C,\infty)}(f_1, g_1) = e^{(C,\infty)}(f_2, g_2)$. ◀

A corollary to Lemma 39 is that if (f_1, g_1) is a Nash equilibrium, then so is (f_2, g_2).

Next, we give a useful characterization of equivalence to stationary strategies.

▶ **Lemma 40.** *Let* $\mathrm{x} \in \Delta_n$. *A randomized strategy* f *is equivalent to* $\mathrm{stat}(\mathrm{x})$ *iff* $f^{(I,J)}(\emptyset) \sim \mathrm{x}$ *for every feasible history* (I, J) *of* f *where* $\mathrm{collisions}(I, J) = 0$.

Proof sketch. (See Appendix C.1 of the full version [23] for the full proof.)

$\Longleftarrow$: For each collision-consistent pair (I, J), where $I := [i_1, \ldots, i_k]$, we show that $\Pr(\mathrm{isResp}(f, I, J)) = \Pr(\mathrm{isResp}(\mathrm{stat}(\mathrm{x}), I, J)) = \prod_{t=1}^{k} \mathrm{x}_{i_t}$.

$\Longrightarrow$: $\forall i, j \in [1..n]$, $\Pr(f^{(I,J)}(\emptyset) = i) = \Pr(\mathrm{isResp}(f, I + [i], J + [j]))/\Pr(\mathrm{isResp}(f, I, J))$. Replace f by $\mathrm{stat}(\mathrm{x})$ (since they're equivalent) and simplify to get $\Pr(f^{(I,J)}(\emptyset) = i) = \mathrm{x}_i$. ◀

Indeed, for any stationary strategy, it's possible to construct a different equivalent strategy. This rules out uniqueness of Nash equilibrium. However, we can still hope to get uniqueness *up to equivalence*. We found this to be a very difficult problem, and could only resolve it for RUC games of the form (A, A) (i.e., zero-sum RUC games), where A is irreducible. For such games, we show that all Nash equilibria are equivalent.

▶ **Theorem 41.** *Let* ρ *be the Perron root of an irreducible matrix* A *and* x^* *and* y^* *be the left and right Perron vectors of* A*, respectively. Then for any Nash equilibrium* (f^*, g^*) *of the RUC game* (A, A)*,* f^* *is equivalent to* $\mathrm{stat}(\mathrm{x}^*)$ *and* g^* *is equivalent to* $\mathrm{stat}(\mathrm{y}^*)$.

Proof sketch. (See Appendix C.3 of the full version [23] for the full proof.)

We first use Proposition 7 to show that (f^*, y^*) and (x^*, g^*) are also NE. If $f^*(\emptyset)$ is not distributed as x^*, then in the NE (f^*, y^*), we show that the min player can decrease her cost by changing her first action's distribution. This contradicts the fact that (f^*, y^*) is an NE, and hence, proves that $f^*(\emptyset) \sim \mathrm{x}^*$. We similarly prove that $g^*(\emptyset) \sim \mathrm{y}^*$.

Next, we show that for any (I, J) such that $\mathrm{collisions}(I, J) = 0$, $({f^*}^{(I,J)}, {g^*}^{(J,I)})$ is also an NE, otherwise the min player can gain by deviating in the NE (f^*, y^*), or the max player can gain by deviating in the NE (x^*, g^*).

Combining the above results tells us that ${f^*}^{(I,J)}(\emptyset) \sim \mathrm{x}^*$ and ${g^*}^{(J,I)}(\emptyset) \sim \mathrm{y}^*$ for all (I, J) such that $\mathrm{collisions}(I, J) = 0$, which fulfills the condition of equivalence in Lemma 40. ◀

5 Discussion

In this work, we initiated the study of two-player RUC games: games that are repeated until collision. RUC games are related to other well-known repeated games, like pursuit-evasion games, hide-and-seek games, and stochastic games. They also generalize the popular game of hand cricket, and in Appendix B of the full version [23], we discuss its popular variants. Our main result showed the existence of Nash equilibria in RUC games when the players' payoff matrices are irreducible. We studied two other interesting properties: *stationarity* and *uniqueness*. We proved there always exists a Nash equilibrium where players use stationary strategies, and for zero-sum RUC games, all Nash equilibria are essentially equivalent, that is, they cannot be distinguished by observing the players' actions.

In Appendix D of the full version [23], we explore a variant of RUC games where instead of ending the game on the first collision, we end it on the wth collision, for some $w \in \mathbb{Z}_{\geq 0}$.

Our work opens the way for several interesting questions. We can naturally generalize the definition of a collision to include a larger set of colliding actions, encoded via a collision matrix, and investigate the existence of Nash equilibria. Lastly, our result showing uniqueness up to equivalence of Nash equilibria applies only to zero-sum RUC games; showing (non-)uniqueness for general RUC games is another open question.

References

1. Franklin Allen and Stephen Morris. Finance applications of game theory. Center for financial institutions working papers, Wharton School Center for Financial Institutions, University of Pennsylvania, 1998. URL: https://EconPapers.repec.org/RePEc:wop:pennin:98-23.
2. Noga Alon, Troy Lee, Adi Shraibman, and Santosh Vempala. The approximate rank of a matrix and its algorithmic applications: Approximate rank. In *Symposium on Theory of Computing (STOC)*, pages 675–684. ACM, 2013. doi:10.1145/2488608.2488694.
3. Shubham Arya. Hand cricket (iOS app). URL: https://theshubhamarya.github.io/HandCricket/.
4. Robert J. Aumann and Lloyd S. Shapley. *Long-Term Competition—A Game-Theoretic Analysis*, pages 1–15. Springer New York, 1994. doi:10.1007/978-1-4612-2648-2_1.
5. Siddharth Barman. Approximating nash equilibria and dense bipartite subgraphs via an approximate version of caratheodory's theorem. In *Symposium on Theory of Computing (STOC)*, pages 361–369. ACM, 2015. doi:10.1145/2746539.2746566.
6. Jean-Pierre Benoit and Vijay Krishna. Finitely repeated games. *Econometrica*, 53(4):905–922, 1985. doi:10.2307/1912660.
7. Richard Borie, Craig Tovey, and Sven Koenig. Algorithms and complexity results for pursuit-evasion problems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 59–66, 2009. URL: https://www.ijcai.org/Proceedings/09/Papers/021.pdf.
8. Dimitris E. Charilas and Athanasios D. Panagopoulos. A survey on game theory applications in wireless networks. *Computer Networks*, 54(18):3421–3430, 2010. doi:10.1016/j.comnet.2010.06.020.
9. Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Computing nash equilibria: Approximation and smoothed complexity, 2006. arXiv:cs/0602043.
10. Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Sparse games are hard. In *Workshop on Internet and Network Economics (WINE)*, pages 262–273. Springer, 2006.
11. Xi Chen and Shang-hua Teng. The approximation complexity of win-lose games. In *Symposium on Discrete Algorithms (SODA)*, pages 159–168. SIAM, 2007.
12. Konstantinos Daskalakis and Christos Papadimitriou. Three-player games are hard. *Electronic Colloquium on Computational Complexity (ECCC)*, 2005.
13. dissknight. How to play hand cricket. https://www.instructables.com/How-to-Play-Hand-Cricket/.
14. James W. Friedman. A non-cooperative equilibrium for supergames. *The Review of Economic Studies*, 38(1):1–12, 1971. doi:10.2307/2296617.
15. Drew Fudenberg and Eric Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica*, 54(3):533–554, 1986. doi:10.2307/1911307.
16. Ravi Kannan and Thorsten Theobald. Games of fixed rank: A hierarchy of bimatrix games. In *Symposium on Discrete Algorithms (SODA)*, pages 1124–1132. SIAM, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283504.
17. Mohammad Emtiyaz Khan. Game theory models for pursuit evasion games, 2007. URL: https://emtiyaz.github.io/Writings/EMTgame.pdf.
18. S.M. LaValle and S. Hutchinson. Game theory as a unifying structure for a variety of robot tasks. In *International Symposium on Intelligent Control*, pages 429–434. IEEE, 1993. doi:10.1109/ISIC.1993.397675.

19 Ruta Mehta. Constant rank bimatrix games are ppad-hard. In *Symposium on Theory of Computing (STOC)*, pages 545–554. ACM, 2014. doi:10.1145/2591796.2591835.

20 Jean-François Mertens, Sylvain Sorin, and Shmuel Zamir. *Repeated Games*. Cambridge University Press, 2015. doi:10.1017/CBO9781139343275.

21 Carl D. Meyer and Ian Stewart. *Matrix Analysis and Applied Linear Algebra, Second Edition*. SIAM, 2023. doi:10.1137/1.9781611977448.

22 R. V. Mises and H. Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM – Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 9(2):152–164, 1929. doi:10.1002/zamm.19290090206.

23 Aniket Murhekar and Eklavya Sharma. Nash equilibria of two-player matrix games repeated until collision, 2023. arXiv:2309.15870v1.

24 John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951. doi:10.2307/1969529.

25 J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928. URL: http://eudml.org/doc/159291.

26 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007. doi:10.1017/CBO9780511800481.

27 Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1994. URL: https://ideas.repec.org/b/mtp/titles/0262650401.html.

28 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.

29 A. Rubinstein. Strong perfect equilibrium in supergames. *International Journal of Game Theory*, 9(1):1–12, 1980. doi:10.1007/BF01784792.

30 Ariel Rubinstein and A. Tversky. Naive strategies in zero-sum games. Working papers, Tel Aviv – The Sackler Institute of Economic Studies, 1993. URL: https://EconPapers.repec.org/RePEc:fth:teavsa:17-93.

31 Ariel Rubinstein, Amos Tversky, and Dana Heller. *Naive Strategies in Competitive Games*, pages 394–402. Springer Berlin Heidelberg, 1997. doi:10.1007/978-3-642-60495-9_30.

32 L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953. doi:10.1073/pnas.39.10.1095.

33 Sylvain Sorin. *A First Course on Zero-Sum Repeated Games*. Springer, 2003.

Synchronized CTL over One-Counter Automata

Shaull Almagor
Department of Computer Science, Technion, Israel

Daniel Assa
Reichman University, Herzliya, Israel

Udi Boker
Reichman University, Herzliya, Israel

Abstract

We consider the model-checking problem of *Synchronized Computation-Tree Logic* (CTL+Sync) over One-Counter Automata (OCAs). CTL+Sync augments CTL with temporal operators that require several paths to satisfy properties in a synchronous manner, e.g., the property "all paths should eventually see p at the same time". The model-checking problem for CTL+Sync over finite-state Kripke structures was shown to be in $\mathsf{P}^{\mathsf{NP}^{\mathsf{NP}}}$. OCAs are labelled transition systems equipped with a non-negative counter that can be zero-tested. Thus, they induce infinite-state systems whose computation trees are not regular. The model-checking problem for CTL over OCAs was shown to be PSPACE-complete.

We show that the model-checking problem for CTL+Sync over OCAs is decidable. However, the upper bound we give is non-elementary. We therefore proceed to study the problem for a central fragment of CTL+Sync, extending CTL with operators that require *all* paths to satisfy properties in a synchronous manner, and show that it is in $\mathsf{EXP}^{\mathsf{NEXP}}$ (and in particular in EXPSPACE), by exhibiting a certain "segmented periodicity" in the computation trees of OCAs.

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

Keywords and phrases CTL, Synchronization, One Counter Automata, Model Checking

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.19

Related Version *Full Version*: `https://arxiv.org/pdf/2308.03308.pdf`

Funding *Shaull Almagor*: supported by the Israel Science Foundation grant 989/22.
Udi Boker: supported by the Israel Science Foundation grant 2410/22.

1 Introduction

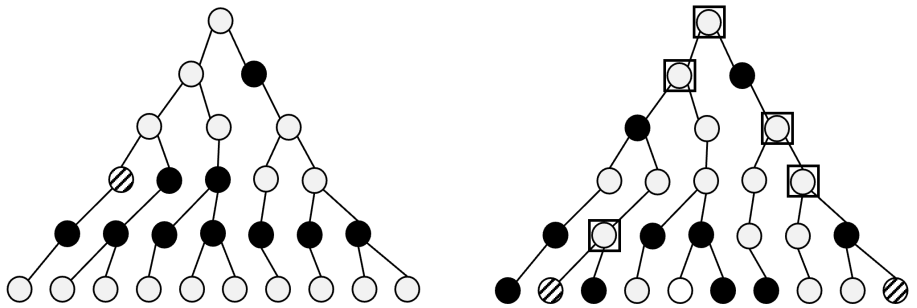
Branching-time model checking is a central avenue in formal verification, as it enables reasoning about multiple computations of the system with both an existential and universal quantification. As systems become richer, the classical paradigm of e.g., CTL model checking over finite-state systems becomes insufficient. To this end, researchers have proposed extensions both of the logics [2, 6, 5, 3, 4, 1] and of the systems [7, 21, 9, 13].

In the systems' frontier, of particular interest are infinite-state models. Typically, such models can quickly lead to undecidability (e.g., two-counter machines [18]). However, some models can retain decidability while still having rich modelling power. One such model that has received a lot of attention in recent years is One Counter Automata (OCAs) [20, 15] – finite state machines equipped with a non-negative counter that can be zero-tested. Model checking CTL over OCAs was studied in [13], where it was shown to be decidable in PSPACE. The main tool used there is the fact that despite the infinite configuration space, the computations of an OCA do admit some periodic behavior, which can be exploited to exhibit a small-model property for the satisfaction of Until formulas.

© Shaull Almagor, Daniel Assa, and Udi Boker;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 19; pp. 19:1–19:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the logics' frontier, a useful extension of CTL is that of CTL with Synchronization operators (CTL+Sync), introduced in [4]. CTL+Sync extends CTL with operators that express synchronization properties of computation trees. Specifically, two new operators are introduced: *Until All* and *Until Exists*. The former, denoted by $\psi_1 U A \psi_2$, holds in state s if there is a uniform bound $k \in \mathbb{N}$ such that ψ_2 holds in all paths from s after exactly k steps, and ψ_1 holds in all paths up to step k. Thus, intuitively, it requires all the computations to synchronize the satisfaction of the Until operator. The latter, somewhat less natural operator, denoted by $\psi_1 U E \psi_2$, requires that there exists a uniform bound k such that in every level $j < k$ of the computation tree, some path satisfies ψ_1 and can be continued to satisfy ψ_2 at level k.

In comparison, the standard CTL operators $A\psi_1 U \psi_2$ and $E\psi_1 U \psi_2$ require that all paths/some path satisfy the Until formula, but there is no requirement that the bounds coincide. We illustrate the differences between the semantics in Figure 1. As discussed in [4], CTL+Sync can describe non ω-regular properties of trees, and hence goes beyond MSO, while retaining a decidable model-checking problem over finite Kripke structures.



(a) $\texttt{true}\, UA$ ● and E ○ U ⊘ are satisfied. **(b)** $A\, \texttt{true}\, U$ ● and ○ UE ⊘ are satisfied.

Figure 1 The computation tree in **(b)** satisfies $A\, \texttt{true}\, U$ ●, since every path eventually reaches ●. However, it does not satisfy $\texttt{true}\, UA$ ●, since this eventuality does not happen synchronously. In contrast, the tree in **(a)** does satisfy $\texttt{true}\, UA$ ●.
The formula E ○ U ⊘ is satisfied in **(a)** (along the left branch), but not in **(b)**, which only satisfies the weaker ○ UE ⊘. The boxed nodes on each level show that ○ holds along a path to ⊘ at level 6.

In this work, we show the decidability of CTL+Sync model checking over OCAs: Given an OCA $\mathcal{A}$ and a CTL+Sync formula φ, the problem of whether $\mathcal{A}$ satisfies φ. We thus combine the expressiveness of CTL+Sync with the rich modeling power of OCAs.

On the technical side, the approach taken in [4] to solve model-checking of CTL+Sync over Kripke structures does not seem to be very useful in our case. The solution there is based on the observation that every two levels of the computation tree that share the same set of Kripke-states must also share the same satisfaction value to every CTL+Sync formula. Hence, in that case, the algorithm can follow the computation tree of the powerset of the given Kripke structure, and terminate when encountering a level that has the same set of states as some previous level. In contrast, for OCAs, the unbounded counter prevents the ability to consider subsets of configurations.

On the other hand, the approach taken in [13] to solve model-checking of OCAs with respect to CTL is indeed useful in our case. Specifically, the algorithm in [13] is based on an analysis of the periodic behavior of the set of counter values that satisfy a given CTL formula in a given state of the OCA. We extend this approach, taking into account the additional complexity that stems from the synchronization requirements; see Section 5.

We start with establishing the decidability of CTL+Sync model checking using Presburger Arithmetic (see Section 4). This, however, yields a procedure with non-elementary runtime.

We then proceed to our main technical contribution (Section 5), providing an algorithm for model checking the central fragment of CTL+Sync that extends CTL with the UA operator, which requires *all* paths to satisfy properties in a synchronous manner. Its running time is in $\mathsf{EXP}^{\mathsf{NEXP}}$ (and in particular in $\mathsf{EXPSPACE}$), and for a fixed OCA and formulas of a fixed nesting depth, it is in P^{NP} (and in particular in PSPACE).

Since CTL+Sync makes assertions on the behavior of different paths at the same time step (namely the same level of the computation tree), we need to reason about which configurations occur at each level of the tree. More precisely, in order to establish decidability we wish to exhibit a small-model property of the form *if the computation tree from some configuration* (s, v)*, for a state* s *and counter value* v*, satisfies the formula* φ*, then the computation tree from some configuration* (s, v') *for a small* v' *satisfies* φ *as well.* Unfortunately, the computation trees of an OCA from two configurations (s, v) and (s, v') cannot be easily obtained from one another using simple pumping arguments, due to the zero tests. This is in contrast to the case where one does not care about the length of a path, as in [13]. To overcome this, we show that computation trees of an OCA $\mathcal{A}$ can be split into several segments, polynomially many in the size of $\mathcal{A}$, and that within each segment we can find a bounded prefix that is common to all trees after a certain counter threshold, and such that the remainder of the segment is periodic. Using this, we establish the small model property above. The toolbox used for proving this, apart from careful cycle analysis, includes 2TVASS – a variant of 2VASS studied in [17] that allows for one counter of the 2VASS to be zero-tested.

We believe that this structural result (Lemma 18) is of independent interest for reasoning about multiple traces in an OCA computation tree, when the length of paths plays a role.

Due to lack of space, Proposition proofs are omitted; they appear in the arXiv version.

2 Preliminaries

Let $\mathbb{N} = \{0, 1, \ldots\}$ be the natural numbers. For a finite set $A \subseteq \mathbb{N}$ we denote by $\mathrm{lcm}(A)$ the least common multiple of the elements in A.

For a finite sequence $\xi = t_0 t_1 \cdots t_{h-1}$ and integers x, y, such that $0 \leq x \leq y \leq h-1$, we write $\xi[x..y]$ for the infix of ξ between positions x and y, namely for $t_x t_{x+1} \cdots t_y$. We also use the parentheses "(" and ")" for a non-inclusive range, e.g., $[x..y) = [x..y-1]$, and abbreviate $\xi[x..x]$ by $\xi[x]$. We denote the length of ξ by $|\xi| = h$.

One Counter Automata

A *One Counter Automaton* (OCA) is a triple $\mathcal{A} = \langle S, \Delta, L \rangle$, where S is a finite set of states, $\Delta \subseteq (S \times \{\texttt{=0}, \texttt{>0}\} \times \{0, +1, -1\} \times S)$ is a transition relation, and $L : S \to AP$, for some finite set AP of *atomic propositions*, is the state labeling.

A pair $(s, v) \in S \times \mathbb{N}$ is a *configuration* of $\mathcal{A}$. We write $(s, v) \to_t (s', v')$ for a transition $t \in \Delta$ if one of the following holds:

- $t = (s, \texttt{=0}, e, s')$, with $e \in \{0, +1\}$, $v = 0$ and $v' = e$, or
- $t = (s, \texttt{>0}, e, s')$, with $e \in \{0, +1, -1\}$, $v > 0$ and $v' = v + e$.

We write $(s, v) \to (s', v')$ if $(s, v) \to_t (s', v')$ for some $t \in \Delta$.

We require that Δ is total, in the sense that for every configuration (s, v) we have $(s, v) \to (s', v')$ for some configuration (s', v'). Note that this is a syntactic requirement – every state should have outgoing transitions on both $\texttt{=0}$ and $\texttt{>0}$. This corresponds to the standard requirement of Kripke structures that there are no deadlocks. We denote by $|\mathcal{A}|$ the number of states of $\mathcal{A}$. Note that the description size of $\mathcal{A}$ is therefore polynomial in $|\mathcal{A}|$.

A *path* of $\mathcal{A}$ is a sequence of transitions $\tau = t_1, \ldots, t_k$ such that there exist states $s_0, \ldots, s_k$ where $t_i = (s_{i-1}, \bowtie_i, e_i, s_i)$ with $\bowtie_i \in \{\texttt{=0}, \texttt{>0}\}$ for all $1 \leq i \leq k$. We say that τ is *valid* from starting counter v_0 (or from configuration (s_0, v_0)), if there are counters $v_0, \ldots, v_k \in \mathbb{N}$ such that for all $1 \leq i < k$ we have $(s_{i-1}, v_{i-1}) \rightarrow_{t_i} (s_i, v_i)$. We abuse notation and refer to the sequence of configurations also as a *path*, starting in (s_0, v_0) and ending in (s_k, v_k). The *length* of the path τ is k, and we define its $\mathit{effect}(\pi) = \sum_{i=1}^{k} e_i$.

We also allow infinite paths, in which case there are no end configurations and the length is ∞. In this case we explicitly mention that the path is infinite. We say that τ is *balanced/positive/negative* if $\mathit{effect}(\tau)$ is zero/positive/negative, respectively. It is a *cycle* if $s_0 = s_k$, and it *has a cycle* β, if β is a cycle and is a contiguous infix of τ.

CTL+Sync

A CTL+Sync formula φ is given by the following syntax, where q stands for an atomic proposition from a finite set AP of atomic propositions.

$$\varphi ::= \underbrace{\texttt{true} \mid q \mid \varphi \wedge \varphi \mid \neg\varphi \mid EX\varphi \mid E\varphi U\varphi \mid A\varphi U\varphi}_{\text{Standard CTL}} \mid \underbrace{\varphi U A\varphi \mid \varphi U E\varphi}_{\text{Sync operators}}$$

We proceed to the semantics. Consider an OCA $\mathcal{A} = \langle S, \Delta, L \rangle$, a configuration (s, v), and a CTL+Sync formula φ. Then $\mathcal{A}^{(s,v)}$ satisfies φ, denoted by $\mathcal{A}^{(s,v)} \models \varphi$, as defined below.

Boolean Opeartors:

- $\mathcal{A}^{(s,v)} \models \texttt{true}$; $\mathcal{A}^{(s,v)} \not\models \texttt{false}$
- $\mathcal{A}^{(s,v)} \models q$ if $q \in L(s)$.
- $\mathcal{A}^{(s,v)} \models \neg\varphi$ if $\mathcal{A}^{(s,v)} \not\models \varphi$.
- $\mathcal{A}^{(s,v)} \models \varphi \wedge \psi$ if $\mathcal{A}^{(s,v)} \models \varphi$ and $\mathcal{A}^{(s,v)} \models \psi$.

CTL temporal operators:

- $\mathcal{A}^{(s,v)} \models EX\varphi$ if $(s, v) \rightarrow (s', v')$ for some configuration (s', v') such that $\mathcal{A}^{(s',v')} \models \varphi$.
- $\mathcal{A}^{(s,v)} \models E\varphi U\psi$ if there exists a valid path τ from (s, v) and $k \geq 0$, such that $\mathcal{A}^{\tau[k]} \models \psi$ and for every $j \in [0..k-1]$, we have $\mathcal{A}^{\tau[j]} \models \varphi$.
- $\mathcal{A}^{(s,v)} \models A\varphi U\psi$ if for every valid path τ from (s, v) there exists $k \geq 0$, such that $\mathcal{A}^{\tau[k]} \models \psi$ and for every $j \in [0..k-1]$, we have $\mathcal{A}^{\tau[j]} \models \varphi$.

Synchronization operators:

- $\mathcal{A}^{(s,v)} \models \varphi U A\psi$ if there exists $k \geq 0$, such that for every valid path τ from (s, v) of length k and for every $j \in [0..k-1]$ we have $\mathcal{A}^{\tau[j]} \models \varphi$ and $\mathcal{A}^{\tau[k]} \models \psi$.
- $\mathcal{A}^{(s,v)} \models \varphi U E\psi$ if there exists $k \geq 0$, such that for every $j \in [0..k-1]$ there exists a valid path τ from (s, v) of length k such that $\mathcal{A}^{\tau[j]} \models \varphi$ and $\mathcal{A}^{\tau[k]} \models \psi$.

▶ Remark 1 (Additional operators). Standard additional Boolean and CTL operators, e.g., $\vee, EF, EG$, etc. can be expressed by means of the given syntax. Similar shorthands can be defined for the synchronization operators, e.g., FE and GE, etc. We remark that one can also consider operators such as $XE\psi$ with the semantics "in the next step there exists a path satisfying ψ". However, the semantics of this coincides with the CTL operator EX.

Presburger Arithmetic

Presburger Arithmetic (PA) [19] is the first-order theory $\mathrm{Th}(\mathbb{N}, 0, 1, +, <, =)$ of $\mathbb{N}$ with addition and order. We briefly survey the results we need about PA, and refer the reader to [14] for a detailed survey.

For our purposes, a PA formula $\varphi(x_1, \ldots, x_d)$, where $x_1, \ldots, x_d$ are free variables, is evaluated over $\mathbb{N}^d$, and *defines* the set $\{(a_1, \ldots, a_d) \in \mathbb{N}^d \mid (a_1, \ldots, a_d) \models \varphi(x_1, \ldots, x_d)\}$. It is known that PA is decidable in 2-NEXP [8, 11].

A *linear set* is a set of the form $\mathrm{Lin}(B, P) = \{\boldsymbol{b} + \lambda_1 \boldsymbol{p_1} + \ldots \lambda_k \boldsymbol{p_k} \mid \boldsymbol{b} \in B,\ \lambda_1 \ldots, \lambda_k \in \mathbb{N}\}$ where $B \subseteq \mathbb{N}^d$ is a finite *basis* and $P = \{\boldsymbol{p_1}, \ldots, \boldsymbol{p_k}\} \subseteq \mathbb{N}^d$ are the *periods*. A *semilinear set* is then a finite union of linear sets. A fundamental result about PA [12] is that the sets definable in PA are exactly the semilinear sets, and moreover, one can effectively obtain from a PA formula φ a description of the semilinear set satisfying a formula φ, and vice versa.

▶ **Observation 2.** *In dimension* 1*, semilinear sets are finite unions of arithmetic progressions. By taking the* lcm *of the periods of the progressions and modifying the basis accordingly, we can assume a uniform period. That is, a semilinear set* $S \subseteq \mathbb{N}$ *is* $\mathrm{Lin}(B, \{p\})$ *for effectively computable* $B \subseteq \mathbb{N}$ *and* $p \in \mathbb{N}$*.*

3 Periodicity and Flatness over OCAs

Recall that the configuration space of an OCA is $S \times \mathbb{N}$. The underlying approach we take to solve CTL+Sync model checking is to show that satisfaction of CTL+Sync formulas over these configurations exhibits some periodicity. Moreover, the run tree of the OCA can be captured, to an extent, using a small number of cycles (a property called *flatness*). These properties will be relied upon in proving our main results.

3.1 Periodicity

In this subsection we formalize our notions of ultimate periodicity, show how they suffice for model-checking, and cite important results about periodicity in CTL.

Consider a CTL+Sync formula φ. We say that φ is $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$*-periodic* with respect to an OCA $\mathcal{A}$ (or just *periodic*, if we do not care about the constants) if for every state $s \in S$ and counters $v, v' > \mathtt{t}(\varphi)$, if $v \equiv v' \mod \mathtt{p}(\varphi)$ then $(s, v) \models \varphi \iff (s, v') \models \varphi$. We think of $\mathtt{t}(\varphi)$ as its threshold and of $\mathtt{p}(\varphi)$ as its period. We say that φ is *totally* $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$*-periodic* with respect to $\mathcal{A}$ if every subformula of φ (including φ itself) is $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$-periodic with respect to $\mathcal{A}$. We usually omit $\mathcal{A}$, as it is clear from context.

Total-periodicity is tantamount to periodicity for each subformula, in the following sense.

▶ **Proposition 3.** *A CTL+Sync formula* φ *is totally* $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$*-periodic if and only if every subformula* ψ *of* φ *is* $(\mathtt{t}'(\psi), \mathtt{p}'(\psi))$*-periodic for some constants* $\mathtt{t}'(\psi), \mathtt{p}'(\psi)$*.*

For a totally periodic formula, model checking over OCA can be reduced to model checking over finite Kripke structures, as follows. Intuitively, we simply "unfold" the OCA and identify states with high counter values according to their modulo of $\mathtt{p}(\varphi)$.

▶ **Proposition 4.** *Consider an OCA* $\mathcal{A}$ *and a totally* $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$*-periodic CTL+Sync formula* φ*, then we can effectively construct a Kripke structure* $\mathcal{K}$ *of size* $|\mathcal{A}| \cdot (\mathtt{t}(\varphi) + \mathtt{p}(\varphi))$ *such that* $\mathcal{A} \models \varphi$ *if and only if* $\mathcal{K} \models \varphi$*.*

In [13, Theorem 1] it is proved that every CTL formula φ over OCA is periodic. Our goal is to give a similar result for CTL+Sync, which in particular contains CTL. In order to avoid replicating the proof in [13] for CTL, we observe that the proof therein is by structural induction over φ, and that moreover – the inductive assumption requires only periodicity of the subformulas of φ. We can thus restate [13, Theorem 1] with the explicit inductive assumption, so that we can directly plug our results about CTL+Sync into it.

Denote by $k = |S|$ the number of states of $\mathcal{A}$, and let $K = \mathrm{lcm}(\{1, \ldots, k\})$.

▶ **Theorem 5** (Theorem 1 in [13], restated). *Consider a CTL+Sync formula φ whose outermost operator is a CTL operator, and whose subformulas are periodic, then φ is periodic, and we have the following.*

- *If $\varphi = \texttt{true}$, $\varphi = \texttt{false}$, or $\varphi = p$ for $p \in AP$, then φ is $(0,1)$-periodic.*
- *If $\varphi = \neg\psi$ then $\texttt{t}(\varphi) = \texttt{t}(\psi)$ and $\texttt{p}(\varphi) = \texttt{p}(\psi)$.*
- *If $\varphi = \psi_1 \wedge \psi_2$ then $\texttt{t}(\varphi) = \max\{\texttt{t}(\psi_1), \texttt{t}(\psi_2)\}$ and $\texttt{p}(\varphi) = \mathrm{lcm}(\texttt{p}(\psi_1), \texttt{p}(\psi_2))$.*
- *If $\varphi = EX\psi$ then $\texttt{t}(\varphi) = \texttt{t}(\psi) + \texttt{p}(\psi)$ and $\texttt{p}(\varphi) = K \cdot \texttt{p}(\psi)$*
- *If $\varphi = E\psi_1 U \psi_2$ or $\varphi = A\psi_1 U \psi_2$ then[1] $\texttt{t}(\varphi) = \max\{\texttt{t}(\psi_1), \texttt{t}(\psi_2)\} + 2 \cdot k^2 \cdot \mathrm{lcm}(K \cdot \texttt{p}(\psi_1), \texttt{p}(\psi_2))$ and $\texttt{p}(\varphi) = \mathrm{lcm}(K \cdot \texttt{p}(\psi_1), \texttt{p}(\psi_2))$.*

3.2 Linear Path Schemes

The runs of an OCA can take intricate shapes. Fortunately, however, we can use the results of [17] about *flatness* of a variant of 2-VASS with some zero tests, referred to as 2-TVASS, to obtain a simple form that characterizes reachability, namely linear path schemes.

A *linear path scheme* (LPS) is an expression of the form $\pi = \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$ where each $\alpha_i \in \Delta^*$ is a path in $\mathcal{A}$ and each $\beta_i \in \Delta^*$ is a cycle in $\mathcal{A}$. The *flat length* of π is $|\pi| = |\alpha_0 \beta_1 \alpha_1 \cdots \beta_k \alpha_k|$, the *size* of π is k.

A concrete path τ in $\mathcal{A}$ is *π-shaped* if there exist $e_1, \ldots, e_k$ such that $\tau = \alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k$.

Our first step is to use a result of [17] on 2-TVASS to show that paths of a fixed length in $\mathcal{A}$ admit a short LPS. The idea is to transform the OCA $\mathcal{A}$ to a 2-TVASS $\mathcal{A}'$ by introducing a length-counting component. That is, in every transition of $\mathcal{A}'$ as a 2-TVASS, the second component increments by 1.

▶ **Lemma 6.** *Let (s, v) and (s', v') be configurations of $\mathcal{A}$. If there exists a path τ of length ℓ from (s, v) to (s', v'), then there is also a π-shaped path τ' of length ℓ from (s, v) to (s', v'), where π is some linear path scheme of flat length poly$(|S|)$ and size $O(|S|^3)$.*

By Lemma 6, we can focus our attention to π-shaped paths where π is "short". Henceforth, we call a path τ *basic* if it is π-shaped for some LPS π as per Lemma 6.

Using standard acceleration techniques (see e.g., [16, 10, 17]), we also get from Lemma 6 that the reachability relation of an OCA (including path length) is effectively semilinear. More precisely, we have the following.

▶ **Corollary 7.** *We can effectively compute, for every $s, s' \in S$, a PA formula $Path_{s,s'}(x, y, x', y')$ such that $(v, \ell, v', \ell') \models Path_{s,s'}(x, y, x', y')$ if and only if a path of length ℓ ending[2] at (s, v) can be continued to a path of length ℓ' ending at (s', v').*

4 Model Checking CTL+Sync via Presburger Arithmetic

In this section we show that model checking CTL+Sync over OCAs is decidable, by reducing it to the satisfiability problem of a PA formula.

We start with a simple observation.

▶ **Lemma 8.** *Consider a totally $(\texttt{t}(\varphi), \texttt{p}(\varphi))$-periodic CTL+Sync formula φ, then for every state $s \in S$ we can compute a PA formula $P_{\varphi,s}(x)$ such $v \models P_{\varphi,s}(x)$ if and only if $(s, v) \models \varphi$.*

---

[1] Note that in [13] the case of $A\psi_1 U \psi_2$ is not stated, but rather the dual Release operator $E\psi_1 R \psi_2$, which follows the same proof.

[2] It is more natural to assume $\ell = 0$ and simply consider paths starting at (s, v). However, our formulation makes things easier later on.

Next, we show that from a PA formula as above we can obtain a threshold and a period.

▶ **Lemma 9.** *Consider a CTL+Sync formula φ and PA formulas $P_{\varphi,s}(x)$, for every state s, such that $v \models P_{\varphi,s}(x)$ iff $(s,v) \models \varphi$. Then φ is $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$-periodic for computable constants $\mathtt{t}(\varphi)$ and $\mathtt{p}(\varphi)$.*

Combining Theorem 5 and Lemma 8 we obtain that every CTL formula (without Sync) can be translated to PA formulas. We now turn to include the Sync operators.

Consider a CTL+Sync formula φ. We construct, by induction on the structure of φ, PA formulas $P_{s,\varphi}(v)$, for every state $s \in S$, such that $P_{s,\varphi}(v)$ holds if and only if $(s,v) \models \varphi$. For the Sync operators, this utilizes the PA formulas of Corollary 7.

- If $\varphi = p$ for an atomic proposition p, then $P_{s,\varphi}(v) = \text{True}$ if s is labeled with p and False otherwise.
- If $\varphi = \neg\psi$, then $P_{s,\varphi}(v) = \neg P_{s,\psi}(v)$.
- If $\varphi = \psi_1 \wedge \psi_2$, then $P_{s,\varphi}(v) = P_{s,\psi_1}(v) \wedge P_{s,\psi_2}(v)$.
- If $\varphi = \mathrm{EX}\psi$, $\varphi = \mathrm{A}\psi_1\mathrm{U}\psi_2$ or $\varphi = \mathrm{E}\psi_1\mathrm{U}\psi_2$ then by the induction hypothesis, ψ, ψ_1 and ψ_2 have corresponding PA formulas, and by Lemma 9 we can compute $\mathtt{t}(\psi)$ and $\mathtt{p}(\psi)$ (and similarly for ψ_1 and ψ_2) such that ψ is $(\mathtt{t}(\psi), \mathtt{p}(\psi))$-periodic. Then, by Theorem 5 we can compute $\mathtt{t}(\varphi), \mathtt{p}(\varphi)$ such that φ is $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$-periodic. Thus, we can apply Lemma 8 to obtain PA formulas for φ.
- If $\varphi = \psi_1\mathrm{UE}\psi_2$, then $P_{s,\varphi}(v) = \exists\ell.\forall\ell' < \ell.\Big(\bigvee_{s'\in S}\Big(\exists v'.\big(Path_{s,s'}(v,0,v',\ell') \wedge P_{s',\psi_1}(v') \wedge \bigvee_{s''\in S}\exists v''.Path_{s',s''}(v',\ell',v'',\ell) \wedge P_{s'',\psi_2}(v'')\big)\Big)\Big)$
- If $\varphi = \psi_1\mathrm{UA}\psi_2$, then $P_{s,\varphi}(v) = \exists\ell.\Big(\Big(\exists v'.\bigvee_{s'\in S}\big(Path_{s,s'}(v,0,v',\ell) \wedge P_{s',\psi_2}(v')\big)\Big) \wedge \Big(\bigwedge_{s'\in S}\forall v'.\big(Path_{s,s'}(v,0,v',\ell) \rightarrow P_{s',\psi_2}(v')\big)\Big) \wedge \forall\ell' < \ell.\Big(\bigwedge_{s'\in S}\forall v'.\big(Path_{s,s'}(v,0,v',\ell') \rightarrow P_{s',\psi_1}(v')\big)\Big)\Big)$

The semantics of the obtained PA formulas match the semantics of the respective Sync operators. By Lemma 9 we now have that for every CTL+Sync formula φ we can compute $\mathtt{t}(\varphi), \mathtt{p}(\varphi)$ such that φ is $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$-periodic. By Proposition 3 we can further assume that φ is totally $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$-periodic. Finally, by Proposition 4 we can decide whether $\mathcal{A} \models \varphi$.

▶ Remark 10 (Complexity). Observe that the complexity of our decision procedure via PA formulas is non-elementary. Indeed, when using a CTL subformula, we translate it, using Lemma 8, to a PA formula that may be exponential in the size of the formula and of the OCA. Thus, we might incur an exponential blowup in every step of the recursive construction, leading to a tower of exponents.

5 Model Checking the CTL+UA Fragment

In this section we consider the fragment of CTL+Sync induced by augmenting CTL with only the Sync operator $\psi_1 UA \psi_2$. For this fragment, we are able to obtain a much better upper bound for model checking, via careful analysis of the run tree of an OCA.

Throughout this section, we fix an OCA $\mathcal{A} = \langle S, \Delta, L\rangle$ with $n = |S| \geq 3$ states, and a CTL+UA formula φ. Consider a configuration (s,v) of $\mathcal{A}$ and a CTL+UA formula $\varphi = \psi_1 UA \psi_2$ with UA being the outermost operator. The satisfaction of φ from (s,v) is determined by the computation tree of $\mathcal{A}$ from (s,v). Specifically, we have that $\mathcal{A}^{(s,v)} \models \varphi$ if there exists some bound $k \in \mathbb{N}$ such that ψ_2 holds in all configurations of level k of the computation tree, and ψ_1 holds in all configurations of levels ℓ for $0 \leq \ell < k$.

Therefore, in order to reason about the satisfaction of φ, it is enough to know which configurations appear in each level of the computation tree. This is in contrast with UE, where we would also need to consider the paths themselves. Fortunately, it means we can use the LPS of Lemma 6 to simplify the proofs.

▶ **Theorem 11.** *Given an OCA $\mathcal{A}$ with $n = |\mathcal{A}|$ and a CTL+UA formula φ, we can compute a counter threshold $\mathtt{cT}$ and a counter period P, both single exponential in n and in the nesting depth of φ, such that φ is $(\mathtt{cT}, \mathsf{P})$-periodic with respect to $\mathcal{A}$.*

Before we delve into the proof of Theorem 11, we show how it implies our main result.

▶ **Theorem 12.** *The model-checking problem for CTL+UA is decidable in $\mathrm{EXP}^{\mathrm{NEXP}}$.*

Proof. Consider a CTL+UA formula φ and an OCA $\mathcal{A}$. By Theorem 11 we can compute $\mathtt{cT}, \mathsf{P}$ single exponential in $|\varphi|$ and $|\mathcal{A}|$, such that φ is $(\mathtt{cT}, \mathsf{P})$-periodic. We then apply Proposition 4 to reduce the problem to model checking φ against a Kripke structure $\mathcal{K}$ of size $|\mathcal{A}| \cdot (\mathtt{cT} + \mathsf{P})$.

Finally, the proof of [4, Theorem 1] shows that model checking the CTL+UA fragment can be done in P^{NP} in the size of the Kripke structure and the formula, yielding an $\mathrm{EXP}^{\mathrm{NEXP}}$ bound in our setting. ◀

▶ Remark 13 (Lower Bounds). The PSPACE-hardness of model checking CTL over OCA from [13] immediately implies PSPACE-hardness in our setting as well. However, tightening the gap between the lower and upper bounds remains an open problem.

The remainder of the paper is devoted to proving Theorem 11.

Cycle Manipulation and Slope Manipulation

A fundamental part of our proof involves delicately pumping and removing cycles to achieve specific counter values and/or lengths of paths. We do this with the following technical tools.

For a path τ, we define the *slope* of τ as $\frac{effect(\tau)}{|\tau|}$. Recall that a basic path is of the form $\tau = \alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k$ adhering to some LPS, where $k = O(|\mathcal{A}|)$ and each α_i and β_i is of length $poly(|\mathcal{A}|)$. We denote by b the maximum flat length of any LPS for a basic path. In particular, b bounds the flat length of the LPS, the size of it, and the length of any cycle or path in it.

We call a cycle *basic* if it is of length at most b. A slope of a path is *basic* if it may be the slope of a basic cycle, namely if it equals $\frac{x}{y}$, where $x \in [-b..b], y \in [1..b]$ and $|x| \leq y$. We denote the basic slopes by $\mathtt{s}_i$, starting with $\mathtt{s}_1$ for the smallest. For example for $b = 3$, the basic slopes are ($\mathtt{s}_1{=}{-1}$, $\mathtt{s}_2{=}{-\frac{2}{3}}$, $\mathtt{s}_3{=}{-\frac{1}{2}}$, $\mathtt{s}_4{=}{-\frac{1}{3}}$, $\mathtt{s}_5{=}0$, $\mathtt{s}_6{=}\frac{1}{3}$, $\mathtt{s}_7{=}\frac{1}{2}$, $\mathtt{s}_8{=}\frac{2}{3}$, $\mathtt{s}_9{=}1$). Observe that for every i, we have $|\mathtt{s}_i| \geq \frac{1}{b}$, and for every $j > i$, we have $\mathtt{s}_j - \mathtt{s}_i \geq \frac{1}{b^2}$ and when they are both negative, also $\frac{\mathtt{s}_j}{\mathtt{s}_i} \leq 1 - \frac{1}{b^2}$.

▶ **Proposition 14.** *Consider a basic path τ and basic cycles c_1, c_2, c_3 in τ with effects $e_1, e_2, e_3 \in [-b..b]$, respectively, and lengths $\ell_1, \ell_2, \ell_3 \in [1..b]$, respectively, such that $\frac{e_1}{\ell_1} \leq \frac{e_2}{\ell_2} \leq \frac{e_3}{\ell_3}$. Then there are numbers $k_1, k_3 \in [0..b^2]$, such that the combination of k_1 repetitions of c_1 and k_3 repetitions of c_3 yield an effect and length whose ratio is $\frac{e_2}{\ell_2}$.*

▶ **Proposition 15.** *Consider a path π with cycles c_1, c_2 with effects $e_1, e_2 \in [-b..b]$ and lengths $\ell_1, \ell_2 \in [1..b]$, respectively, such that $\frac{e_1}{\ell_1} < \frac{e_2}{\ell_2}$, and a length x that is divisible by $\mathrm{lcm}[1..2b^2]$. Then there are numbers $k_1, k_2 \in [0..b \cdot x]$, such that the addition or removal of k_1 repetitions of c_1 and the addition or removal of k_2 repetitions of c_2 yield a path of the same effect as π and of a length shorter or longer, as desired, by x (provided enough cycle repetitions exist).*

In order to prove Theorem 11, we show that every computation tree of $\mathcal{A}$, starting from a big enough counter value $v > \mathtt{cT}$, has a "segmented periodic' structure with respect to φ. That is, we can divide its levels into $poly(n)$ many *segments*, such that only the first $\mathtt{sT} \in Exp(n, |\varphi|)$ levels in each segment are in the "core', while every other level ℓ is a sort-of repetition of the level $\ell - \mathtt{P}$, for a *period* $\mathtt{P}$. We further show that there is a similarity between the cores of computation trees starting with counter values v and $v + \mathtt{P}$. We depict the segmentation in Figure 2, and formalize it as follows.

Consider a formula $\varphi = \psi_1 U A \psi_2$, where ψ_1 and ψ_2 are $(\mathtt{cT}(\psi_1), \mathtt{P}(\psi_1))$- and $(\mathtt{cT}(\psi_2), \mathtt{P}(\psi_2))$-periodic, respectively. We define several constants to use throughout the proof.

Constants depending only on the number n of states in the OCA

- $b \in Poly(n)$: the bound on the length of a linear path scheme on $\mathcal{A}$.
- $B = \mathrm{lcm}[1..2b^3]$.

Constants depending on n and the CTL+UA formula φ

- $\mathtt{P}_{prev}(\varphi) = \mathrm{lcm}(\mathtt{P}(\psi_1), \mathtt{P}(\psi_2))$ – the unified period of the subformulas.
- $\mathtt{cT}_{prev}(\varphi) = \max(\mathtt{cT}(\psi_1), \mathtt{cT}(\psi_2))$ – the unified counter threshold of the subformulas.
- $\mathtt{P}(\varphi) = B \cdot \mathtt{P}_{prev}(\varphi)$ – the "period" of φ.
- $\mathtt{sT}(\varphi) = b^9 \cdot \mathtt{P}(\varphi)$ – the "segment threshold" of φ.
- $\mathtt{cT}(\varphi) = b^{11} \cdot \mathtt{P}(\varphi)$ – the "counter threshold" of φ.

Eventually, these periodicity constants are plugged into the inductive cases of Theorem 5, as shown in the proof of Theorem 11. Then, all the constants are single-exponential in n and the nesting depth of φ. Notice that the following relationship holds between the constants.

▶ **Proposition 16.** *$P(\varphi) > cT_{prev}(\varphi)$ for $b \geq 2$.*

When clear from the context, we omit the parameter φ from $\mathtt{P}, \mathtt{sT}, \mathtt{cT}, \mathtt{P}_{prev}, \mathtt{cT}_{prev}$.

We provide below an intuitive explanation for the choice of constants above.

Intuition for the period P

The period has two different roles: *level-periodicity* within each segment of a computation tree, and *counter-value periodicity* between two computation trees starting with different counter values.

Level periodicity within a segment: For lengthening or shortening a basic path by $\mathtt{P}$, we add and/or remove some copies of its cycles. Adding or removing $\mathtt{P}_{prev}$ copies of the same cycle guarantees that the end counter values of the original and new paths are equivalent modulo $\mathtt{P}_{prev}$. Since the cycles in a basic path are of length in $[0..b]$, setting $\mathtt{P}$ to be divisible by $\mathtt{P}_{prev} \cdot \mathrm{lcm}[1..b]$, allows to add or remove $\frac{\mathtt{P}}{|c|}$ copies of a cycle c, where $\frac{\mathtt{P}}{|c|}$ is divisible by $\mathtt{P}_{prev}$, as desired. Yet, we might need to add copies of one cycle and remove copies of another, thus, as per Proposition 15, we need $\mathtt{P}$ to be divisible by $\mathtt{P}_{prev} \cdot lcm[1..2b^2]$.

Counter periodicity between computation trees: We change a path τ that starts with a counter value v to a path that starts with a counter value $v + \mathtt{P}$, or vice versa, by lengthening or shortening it by $\frac{\mathtt{P}}{\mathtt{s}}$, respectively, where $\mathtt{s}$ is a positive basic slope. In some cases, we need to also make sure that the longer or shorter path has a drop bigger or smaller, respectively, than τ by exactly $\mathtt{P}$.

As $\frac{\mathtt{P}}{\mathtt{s}}$ is bounded by $b \cdot \mathtt{P}$, if there are at least $b \cdot \mathtt{P}$ repetitions of a cycle c in τ whose slope is $-\mathtt{s}$, we can just add or remove $\frac{b \cdot \mathtt{P}}{|c|}$ copies of c, so we need $\mathtt{P}$ to be divisible by $\mathtt{P}_{prev} \cdot \mathrm{lcm}[1..b]$, for guaranteeing that the counter values at the end of the original and new paths are equivalent. Yet, in some cases we need to combine two cycles, as per Proposition 14. As the combination of the two cycles might be of length up to $2b^3$, we need $\mathtt{P}$ to be divisible by $\mathtt{P}_{prev} \cdot \mathrm{lcm}[1..2b^3]$.

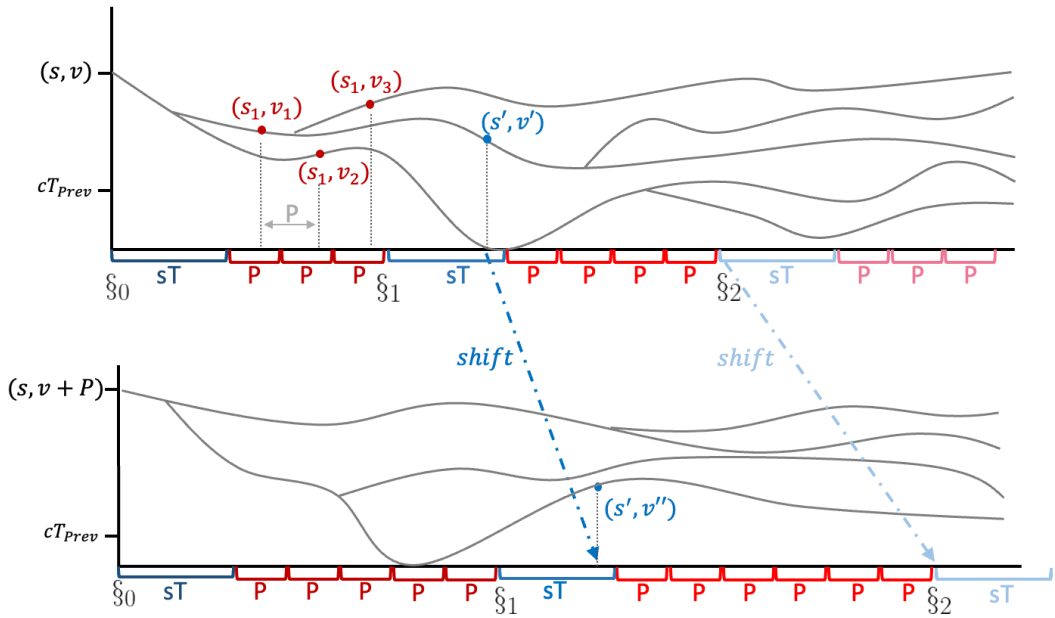


Figure 2 A computation tree from (s, v) above and from $(s, v + \mathtt{P})$ below, demonstrating the *segmented periodicity*. Each segment i starts at $\S_i$ and ends at $\S_{i+1} - 1$ (except for the last, which never ends). The *core* of the computation tree is the union of the first $\mathtt{sT}$ positions from each segment. Following these $\mathtt{sT}$ positions, there is within each segment periodicity of length $\mathtt{P}$, meaning that for every path of length ℓ, there exists an equivalent path of length $\ell + \mathtt{P}$, as shown for the three points with the same state s and equivalent counter values $v_1 \equiv v_2 \equiv v_3$. Between the trees, there is equivalence between the core positions, mapped via the shift function, as shown for the two points with the same state s' and the equivalent counter values $v' \equiv v''$.

Intuition for the counter threshold $\mathtt{cT}$ and the segment threshold $\mathtt{sT}$

In order to apply Propositions 14 and 15, we need to have in the handled path many repetitions of two cycles of different slopes. We thus choose $\mathtt{cT}$ and $\mathtt{sT}$ to be large enough so that paths in which only one (negative) cycle slope is repeated many times must hit zero within a special region called the "core" of the tree, as defined below.

The core of a computation tree

For every counter value $v > \mathtt{cT}$, the "core" with respect to a fixed formula φ, denoted by $\mathtt{core}(v) \subseteq \mathbb{N}$, of a computation tree of $\mathcal{A}$ that starts with a counter value v consists of $m + 1 < b^2$ segments, with each segment corresponding to a negative basic slope and having $\mathtt{sT}$ consequent numbers. For every $i \in [0..m]$, the start of Segment i depends on the initial counter value v of the computation tree, it is denoted by $\S_i(v)$, and it is defined as follows:

- $\S_0(v) = 0$,
- For $i \in [1..m]$, we set $\S_i(v) = \frac{-1}{\mathtt{s}_i}(v - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P}$.
- For convenience, we also define $\S_{m+1}(v) = \infty$.

Then, we define $\mathtt{core}(v) = \bigcup_{i=0}^{m} [\S_i(v)..(\S_i(v) + \mathtt{sT} - 1)]$.

Observe that the core of every tree is an ordered list of $((m + 1) \cdot \mathtt{sT})$ numbers (levels), while just the starting level of every segment depends on the initial counter value v. We can thus define a bijection $\mathsf{shift} : \mathtt{core}(v) \to \mathtt{core}(v + \mathtt{P})$ that maps the i-th number in $\mathtt{core}(v)$ to the i-th number in $\mathtt{core}(v + \mathtt{P})$ (see also Figure 2).

Recall that we define $\mathtt{sT}$ so that, intuitively, if a path is long enough to reach $\S_i(v) + \mathtt{sT}$ without reaching counter value $\mathtt{cT}_{prev}$, then the path must have many cycles with a slope larger than $\mathtt{s}_i$, and if the path manages to reach $\mathtt{cT}_{prev}$ before $\S_{i+1}(v)$ (namely the end of Segment i), then it must have many cycles with a slope at least as small as $\mathtt{s}_i$. This is formalized as follows.

▶ **Lemma 17.** *Let τ be a basic path, or a prefix of it, of length ℓ starting from counter value $v > \mathtt{cT}$, and let $i \in [0..m]$.*

1. *If $\ell \geq \S_i(v) + \frac{\mathtt{sT}}{2}$ and the counter values of τ stay above cT_{prev}, then τ has a cycle with slope $\mathtt{s}_j$ for $j > i$ that repeats at least $b^4 \cdot \mathtt{P}$ times.*
2. *If $\ell < \S_{i+1}(v)$ and the counter values of τ reach cT_{prev}, then τ has a cycle with slope $\mathtt{s}_j$ for $j \leq i$ that repeats at least $b^4 \cdot \mathtt{P}$ times.*

As a sanity check, the lemma states that if a path τ reaches $\mathtt{cT}_{prev}$ for the first time at length $\ell \in [\S_1(v)+\frac{\mathtt{sT}}{2} \,..\, \S_2(v))$, then it has many cycles with slope $\mathtt{s}_1 = -1$ (enough to decrease down to $\mathtt{cT}_{prev}$ before $\S_2(v)$), as well as many cycle with slope at least $\mathtt{s}_2 = \frac{-(b-1)}{b}$ (enough to keep above $\mathtt{cT}_{prev}$ through $\S_1(v) + \frac{\mathtt{sT}}{2}$). Observe also that a path cannot reach $\mathtt{cT}_{prev}$ at Segment 0, namely before $\S_1(v)$.

The segment and shift periodicity

Consider a threshold T and period P. We say that counter values u, v are (T, P)-*equivalent*, denoted by $u \equiv_{T,P} v$ if either $u, v \geq T$ and P divides $|u - v|$, or $u, v < T$ and $u = v$. That is, either both u, v are greater than T, in which case they are equivalent modulo P, or they are both smaller than T and are equal.

The segment periodicity within a computation tree is then stated as Claim 1 in Lemma 18 below, while the similarity between computation trees starting from counters v and $v + \mathtt{P}$ as Claim 2. (By $(s, v) \overset{\ell}{\rightsquigarrow} (s', v')$ we mean that the computation tree starting with state s and counter value v has a path of length ℓ ending in state s' and counter value v'.)

▶ **Lemma 18.** *Consider states s and e, a counter value $v > \mathtt{cT}$, an arbitrary counter value u, and an arbitrary path length ℓ.*

1. *If $\ell \notin \mathtt{core}(v)$ then:*
   - **a.** $(s, v) \overset{\ell}{\rightsquigarrow} (e, u) \implies (s, v) \overset{\ell - P}{\rightsquigarrow} (e, u')$, *and*
   - **b.** $(s, v) \overset{\ell - P}{\rightsquigarrow} (e, u) \implies (s, v) \overset{\ell}{\rightsquigarrow} (e, \tilde{u})$,

   for some counter values u' and $\tilde{u}$, such that $u \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} u' \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} \tilde{u}$.
2. *If $\ell \in \mathtt{core}(v)$ then:*
   - **a.** $(s, v + P) \overset{\mathit{shift}(\ell)}{\rightsquigarrow} (e, u) \implies (s, v) \overset{\ell}{\rightsquigarrow} (e, u')$, *and*
   - **b.** $(s, v) \overset{\ell}{\rightsquigarrow} (e, u) \implies (s, v + P) \overset{\mathit{shift}(\ell)}{\rightsquigarrow} (e, \tilde{u})$

   for some counter values u' and $\tilde{u}$, such that $u \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} u' \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} \tilde{u}$.

Proof. Throughout the proof, we will abbreviate $u \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} u'$ by $u \equiv u'$. We split the proof into four parts, each devoted to one of the four stated implications. In each of them, we assume the existence of a path τ that witnesses the left side of the implication, say $(s, v) \overset{\ell}{\rightsquigarrow} (e, u)$, and show that there exists a path τ' that witnesses the right side of the implication, say $(s, v) \overset{\ell - \mathtt{P}}{\rightsquigarrow} (e, u')$, where $u \equiv u'$. By Lemma 6, we assume that τ is a basic path. We present some of the cases; the remaining parts are in the appendix.

Proof of Lemma 18.1a

Let τ be a basic path of length $\ell \notin \texttt{core}(v)$ such that $(s,v) \overset{\ell}{\rightsquigarrow} (e,u)$ via τ. We construct from τ a path τ' for $(s,v) \overset{\ell-\texttt{P}}{\rightsquigarrow} (e,u')$, such that $u \equiv u'$. The proof is divided to two cases.

Case 1a.1: The counter values in τ stay above $\texttt{cT}_{prev}$

If there is no position in τ with counter value $\texttt{cT}_{prev}$, then in particular τ has no zero-transitions. Since $\ell \notin \texttt{core}(v)$, then in particular $\ell \geq \texttt{sT} > 3b^5\texttt{P}$. Thus, there are at least $3b^4\texttt{P}$ cycle repetitions in τ.

If there is a non-positive cycle c that is repeated at least $\texttt{P}$ times, we can obtain τ' by removing $\frac{\texttt{P}}{|c|}$ copies of c, as the counter values along τ' are at least as high as the corresponding ones in τ. Observe that τ' is of length $\ell - \texttt{P}$ from (s,v) to (e,u') with $u' = u - \mathit{effect}(c)\frac{\texttt{P}}{|c|}$. Since we have $u' \geq u \geq \texttt{cT}_{prev}$, then $u \equiv u'$.

Otherwise, each non-positive cycle in τ is taken at most $\texttt{P}$ times. Thus, the positive cycles are repeated at least $3b^4\texttt{P} - b\texttt{P} \geq 3b^3\texttt{P}$ times. In particular, there exists a positive cycle c that repeats at least $3b^2\texttt{P}$ times. By removing $\frac{\texttt{P}}{|c|}$ occurrences of it, we obtain a path τ' of length $\ell - \texttt{P}$. Notice first that this path is valid. Indeed, up until the cycle c is taken, the path τ' coincides with τ, so the counter remains above $\texttt{cT}_{prev}$. Since c is a positive cycle, after completing its iterations, the counter value becomes at least $3b^2\texttt{P} - \texttt{P} + \texttt{cT}_{prev}$. Then, even if all remaining transitions in the negative cycles have effect -1, the counter value is reduced by at most $b^2\texttt{P}$ (as there are at most $(b-1)\texttt{P}$ remaining cycles, each of effect at least $-b$, and the simple paths in τ can reduce by another b at most). Thus, the value of the counter remains at least $3b^2\texttt{P} - \texttt{P} + \texttt{cT}_{prev} - b^2\texttt{P} > \texttt{cT}_{prev}$. Finally, let (e,u') be the configuration reached at the end of τ', then $u - u' = \mathit{effect}(c)\frac{\texttt{P}}{|c|}$, so $u \equiv u'$.

Case 1a.2: τ reaches counter value $\texttt{cT}_{prev}$

Let $0 \leq z_f \leq z_u \leq \ell$ be the first and ultimate positions in τ where the counter value is exactly $\texttt{cT}_{prev}$. We split τ into three parts: $\tau_1 = \tau[0 \,..\, z_f), \tau_2 = \tau[z_f \,..\, z_u), \tau_3 = \tau[z_u \,..\, \ell]$ (it could be that $z_f = z_u$, in which case the middle part is empty). Since τ is of length $\ell \geq \texttt{sT} \geq b^9 \cdot \texttt{P}$, then at least one of the parts above is of length at least $b^8 \cdot \texttt{P}$ (recall $b \geq 3$). We split according to which part that is. For simplicity, we start with the cases that τ_2 or τ_3 are long, and only then handle the case of a long τ_1.

1. *The middle part $\tau_2 = \tau[z_f \,..\, z_u]$ is of length at least $b^8\texttt{P}$.*
   As τ_2 is of length at least $b^8 \cdot \texttt{P}$, some cycle c in it must repeat at least $b^6 \cdot \texttt{P}$ times.
   If c is balanced, we can obtain τ' by removing $\frac{\texttt{P}}{|c|}$ of its repetitions.
   If c is positive, starting at position x with counter value v_x, then the counter value at position y where c's repetitions end is at least $v_x + b^6 \cdot \texttt{P}$. As τ_2 eventually gets down to $\texttt{cT}_{prev} < \texttt{P}$, there must be a negative cycle c_- that repeats at least $b \cdot \texttt{P}$ times between position y and the first position after y that has the counter value $v_x + \frac{b^6 \cdot \texttt{P}}{2}$. Hence, we can obtain τ' by removing repetitions of c and c_-, as per Proposition 15, ensuring that the only affected values are above v_x.
   If c is negative, starting at position x with counter value v_x, then $v_x \geq b^6 \cdot \texttt{P}$. As τ_2 starts with counter value $\texttt{cT}_{prev}$, and $\texttt{cT}_{prev} < P$ (Proposition 16), there must be a positive cycle c_+ that repeats at least $b \cdot \texttt{P}$ times between the last position with counter value $\frac{b^6 \cdot \texttt{P}}{2}$ and x. Hence, we can obtain τ' by removing repetitions of c and c_+, as per Proposition 15, ensuring that the only affected values are above $b^5 \cdot \texttt{P}$.

2. *The last part $\tau_3 = \tau[z_u \mathinner{..} \ell]$ is of length at least $b^8\mathtt{P}$.*
   As in the previous case, a cycle c must repeat in τ_3 at least $b^6 \cdot \mathtt{P}$ times. If c is balanced, we can remove $\frac{\mathtt{P}}{|c|}$ of its repetitions, getting the desired path τ'.
   Otherwise, it must be that τ_3 stays above $\mathtt{cT}_{prev}$, and reaches a value at least $b^6 \cdot \mathtt{P}$. Indeed, if c is positive then its repetitions end at some position x with a counter value at least that high, and if it is negative it starts at some position x with a counter value at least that high.
   If the counter value also drops to $\frac{b^6 \cdot \mathtt{P}}{2}$ after position x, then we can remove positive and negative cycles exactly as in the previous case. Otherwise, we can just remove $\frac{\mathtt{P}}{|c|}$ repetitions of c, guaranteed that the counter value at the end of τ_3 is above $\mathtt{cT}_{prev}$.
3. *Only the first part $\tau_1 = \tau[0 \mathinner{..} z_f)$ is of length at least $b^8\mathtt{P}$.*
   If any of the other parts is long, we shorten them. Otherwise, their combined length is less than $2b^8 \cdot \mathtt{P} < \frac{\mathtt{sT}}{2}$, implying that the first part τ_1 is longer than $\S_i(v) + \frac{\mathtt{sT}}{2}$.
   Hence, by Lemma 17, there are "fast" and "slow" cycles c_f and c_s, respectively, of slopes $\mathtt{s}_f < \mathtt{s}_s$, such that each of them repeats at least $b^4 \cdot \mathtt{P}$ times in τ_1.
   Thus, by Proposition 15, we can add and/or remove some repetitions of c_f and c_s, such that τ_1 is shorten by exactly $\mathtt{P}$. Yet, we should ensure that the resulting path τ' is valid, in the sense that its corresponding first part τ_1' cannot get the counter value to 0. We show it by cases:
   - If c_f or c_s are balanced cycles, then we can remove the balanced cycle only, without changing the remaining counter values.
   - If there is a positive cycle c_+ that repeats at least $2b^2 \cdot \mathtt{P}$ times, then the counter value climbs by at least $2b^2 \cdot \mathtt{P}$ from its value v_x at position x where c_+ starts and the position y where its repetitions end. As the counter gets down to $\mathtt{cT}_{prev}$ at the end of τ_1, and $\mathtt{cT}_{prev} < P$ (Proposition 16), there must be a negative cycle c_- that repeats at least $b \cdot \mathtt{P}$ times between position y and the first position after y that has the counter value $v_x + b^2 \cdot \mathtt{P}$. Hence, we can remove repetitions of c_+ and c_-, as per Proposition 15, ensuring that the only affected values are above v_x.
   - Otherwise, both c_f and c_s are negative, implying that we add some repetitions of c_f and remove some repetitions of c_s. We further split into two subcases:
     - If c_s appears before c_f then there is no problem, as the only change of values will be their increase, and all the values were nonzero to begin with (as we are before z_f).
     - If c_f appears first, ending at some position x, while c_s starts at some later position y, then a-priori it might be that repeating c_f up to $b \cdot \mathtt{P}$ more times, as per Proposition 15, will take the counter value to 0.
       However, observe that since there are at most $b-2$ positive cycles, and each of them can repeat at most $2b^2 \cdot \mathtt{P} - 1$ times, the counter value v_x at position x, and along the way until position y, is at least $v_y - (b-2)2b^2 \cdot \mathtt{P}$, where v_y is the counter value at position y. As c_s repeats at least $b^4 \cdot \mathtt{P}$ times, we have $v_y \geq b^4 \cdot \mathtt{P}$. Thus $v_x \geq b^4 \cdot \mathtt{P} - 2(b-2)b^2 \cdot \mathtt{P} > b^2\mathtt{P}$. Hence, repeating c_f up to $b \cdot \mathtt{P}$ more times at position x cannot take the counter value to 0, until position y, as required.

Proof of Lemma 18.2a

The case of Segment $\S_0$

For a path of length ℓ, we have in Segment 0 that $\mathsf{shift}(\ell) = \ell$, and indeed a path from v is valid from $v + \mathtt{P}$ and vise versa, as they do not hit $\mathtt{cT}_{prev}$: Their maximal drop is $\mathtt{sT}$, while $v \geq \mathtt{cT} > \mathtt{P} + \mathtt{sT} > \mathtt{cT}_{prev} + \mathtt{sT}$.

We turn to the ith segment, for $i \geq 1$. Consider a basic path τ for $(s, v + \mathtt{P}) \overset{\mathsf{shift}(\ell)}{\rightsquigarrow} (e, u)$. Recall that $\mathsf{shift}(\ell) = \ell + \frac{\mathtt{P}}{-\mathtt{s}_i} \in [\S_i(v + \mathtt{P}) \mathinner{..} \S_i(v + \mathtt{P}) + \mathtt{sT})]$. We construct from τ a path τ' for $(s, v) \overset{\ell}{\rightsquigarrow} (e, u')$, such that $u \equiv u'$, along the following cases.

Case 2a.1: The counter values in τ stay above $\mathtt{cT}_{prev}$

As there is no position in τ with counter value $\mathtt{cT}_{prev}$, then in particular τ has no zero-transitions. We further split into two subcases:

1. If τ does not have $b \cdot \mathtt{P}$ repetitions of a "relatively fast" cycle with slope $\mathtt{s}_j$ for $j \leq i$, then the drop of τ, and of every prefix of it, is at most $X + Y$, where X stands for the drop outside "slow" cycles of slope $\mathtt{s}_h$ for $h > i$, and Y for the rest of the drop. We have $X < b^3 \cdot \mathtt{P}$ and $Y < (\ell + \frac{\mathtt{P}}{\mathtt{s}_i})(-\mathtt{s}_{i+1})$.
   We claim that we can obtain τ' by removing $\frac{\mathtt{P}}{-\mathtt{s}_i \cdot |c|}$ repetitions of any cycle c, which repeats enough in τ, having that the drop D of τ' is less than $v - \mathtt{cT}_{prev}$.
   Indeed, the maximal such drop D might be the result of removing only cycles of slope $(+1)$, whose total effect is $\frac{\mathtt{P}}{-\mathtt{s}_i}$, having $D \leq \frac{\mathtt{P}}{-\mathtt{s}_i} + X + Y = \frac{\mathtt{P}}{-\mathtt{s}_i} + b^3 \cdot \mathtt{P} + (\ell + \frac{\mathtt{P}}{-\mathtt{s}_i})(-\mathtt{s}_{i+1}) \leq b \cdot \mathtt{P} + b^3 \cdot \mathtt{P} + (\ell + \frac{\mathtt{P}}{-\mathtt{s}_i})(-\mathtt{s}_{i+1})$. Since $\ell < \S_{i+1}(v + \mathtt{P}) = \frac{1}{-\mathtt{s}_{i+1}}(v + \mathtt{P} - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P}$, we have $D \leq (b^3 + b) \cdot \mathtt{P} + (\frac{1}{-\mathtt{s}_{i+1}}(v + \mathtt{P} - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P}) + \frac{\mathtt{P}}{-\mathtt{s}_i})(-\mathtt{s}_{i+1}) =$
   $(b^3 + b) \cdot \mathtt{P} + v + \mathtt{P} - \mathtt{cT}_{prev} - (-\mathtt{s}_{i+1}) \cdot b^8 \cdot \mathtt{P} + \frac{(-\mathtt{s}_{i+1})}{-\mathtt{s}_i} \cdot \mathtt{P}) = (b^3 + b + 1 + \frac{(-\mathtt{s}_{i+1})}{-\mathtt{s}_i}) \cdot \mathtt{P} + v - \mathtt{cT}_{prev} - (-\mathtt{s}_{i+1}) \cdot b^8 \cdot \mathtt{P}) < b^4 \cdot \mathtt{P} - b^7 \cdot \mathtt{P} + v - \mathtt{cT}_{prev}$. It is thus left to show that $b^4 \cdot \mathtt{P} < b^7 \cdot \mathtt{P}$, which obviously holds.
2. Otherwise, namely when τ does have $b \cdot \mathtt{P}$ repetitions of a "relatively fast" cycle with slope $\mathtt{s}_j$ for $j \leq i$, let c be the first such cycle in τ. We can obtain τ' by removing $\frac{\mathtt{P}}{-\mathtt{s}_i \cdot |c|}$ repetitions of c: The counter value in τ', which starts with counter value v, at the position after the repetitions of c will be at least as high as the counter value in τ, which starts with counter value $v + \mathtt{P}$, after the repetitions of c. Notice that the counter value cannot hit $\mathtt{cT}_{prev}$ before arriving to the repetitions of c by the argument of the previous subcase.

Case 2a.2: τ reaches counter value $\mathtt{cT}_{prev}$

Again let $\tau_1 = \tau[0 \mathinner{..} z_f), \tau_2 = \tau[z_f \mathinner{..} z_u), \tau_3 = \tau[z_u \mathinner{..} \mathsf{shift}(\ell)]$ as in 1a.

In order to handle possible zero transitions, we shorten τ_1, such that the resulting first part τ_1' of τ', which starts with counter value v, also ends with counter value exactly $\mathtt{cT}_{prev}$. Since τ_1 reaches $\mathtt{cT}_{prev}$ and is shorter than $\S_{i+1}(v + \mathtt{P})$, it has by Lemma 17.2 at least $b^4 \cdot \mathtt{P}$ repetitions of a "fast" cycle of slope $\mathtt{s}_f \leq \mathtt{s}_i$. Let c_f be the first such cycle. We split to cases.

1. If $\mathtt{s}_f = \mathtt{s}_i$ or τ_2 or τ_3 are of length at least $b^5 \cdot \mathtt{P}$, we can remove $\frac{\mathtt{P}}{-\mathtt{s}_f \cdot |c_f|}$ repetitions of c_f in τ_1. Note that the resulting first part τ_1' of τ' indeed ends with counter value $\mathtt{cT}_{prev}$. However, while when $\mathtt{s}_f = \mathtt{s}_i$ the resulting length of τ' will be ℓ, as required, in the case that $\mathtt{s}_f < \mathtt{s}_i$, we have that τ' will be longer than ℓ. Nevertheless, in this case, as τ_2 or τ_3 are of length at least $b^5 \cdot \mathtt{P}$, we can further shorten τ_2 or τ_3 without changing their effect, as per Proposition 15, analogously to 1 or 2, respectively, in the proof of Lemma 18.1a.2.
2. Otherwise, we are in the case that τ_1 has a "really fast" cycle of slope $\mathtt{s}_f < \mathtt{s}_i$ that repeats at least $b^4 \cdot \mathtt{P}$ times, and both τ_2 or τ_3 are of length less than $b^5 \cdot \mathtt{P}$. We claim that in this case τ_1 must also have $b^4 \cdot \mathtt{P}$ repetitions of a "relatively slow" cycle c_s of slope $\mathtt{s}_s \geq \mathtt{s}_i$.
   Indeed, assume toward contradiction that τ_1 has less than $b^4 \cdot \mathtt{P}$ repetitions of a cycle with slope $\mathtt{s}_s$ for $s \geq i$. Then the longest such path has less than b transitions of $(+1)$ out of cycles, $b^6 \cdot \mathtt{P}$ such transitions in cycles, and the rest of it consists of "fast" cycles with slope indexed lower than i.

Thus its length is at most $X + L$, where $X = b^6 \cdot \mathtt{P}$ is the $(+1)$ transitions, and L is the longest length to drop from counter value $v + \mathtt{P} + X$ to $\mathtt{cT}_{prev}$ with "fast" cycles. Thus, $L \leq \frac{1}{-\mathtt{s}_{i-1}}(v + \mathtt{P} + X - \mathtt{cT}_{prev})$.

Now, we have that the length of τ is at least $\S_i(v + \mathtt{P}) = \frac{1}{-\mathtt{s}_i}(v - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P}$. Thus, parts τ_2 and τ_3 of τ are of length at least $Z = \frac{1}{-\mathtt{s}_i}(v - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P} - \frac{1}{-\mathtt{s}_{i-1}}(v + \mathtt{P} + X - \mathtt{cT}_{prev}) - X = (\frac{1}{-\mathtt{s}_i} - \frac{1}{-\mathtt{s}_{i+1}})(v - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P} - (1 + \frac{1}{-\mathtt{s}_{i+1}})b^6 \cdot \mathtt{P}$.

Since $(\frac{1}{-\mathtt{s}_i} - \frac{1}{-\mathtt{s}_{i+1}}) \geq \frac{1}{b^2}$, $(1 + \frac{1}{-\mathtt{s}_{i+1}} \leq b+1)$, and $v - \mathtt{cT}_{prev} \geq \mathtt{cT} - \mathtt{cT}_{prev} > \mathtt{cT} - \mathtt{P} > 3b^{10}\mathtt{P}$, we have $Z \geq \frac{1}{b^2}(3b^{10} \cdot \mathtt{P}) - b^8 \cdot \mathtt{P} - (b+1)b^6 \cdot \mathtt{P} = 2b^8 \cdot \mathtt{P} - (b+1)b^6 \cdot \mathtt{P}$. Therefore, at least one of τ_2 and τ_3 is of length at least $b^7 \cdot \mathtt{P}$, leading to a contradiction.

So, we are in the case that τ_1 has at least $b^4 \cdot \mathtt{P}$ repetitions of a "really fast" cycle c_f of slope $\mathtt{s}_f < \mathtt{s}_i$ as well as $b^4 \cdot \mathtt{P}$ repetitions of a "relatively slow" cycle c_s of slope $\mathtt{s}_s \geq \mathtt{s}_i$. By analyzing the different possible orders of c_s and c_f, we can cut and repeat the cycles far enough from 0 so as to construct valid paths. See the Appendix for details. ◀

5.1 Proof of Theorem 11

Proof. The proof is by induction over the structure of φ, where Theorem 5 already provides the periodicity for all CTL operators.

It remains to plug UA into the induction by showing (1) the $(\mathtt{cT}, \mathtt{P})$-periodicity of a formula $\varphi = \psi_1 UA \psi_2$ with respect to an OCA $\mathcal{A}$, provided that its subformulas are $(\mathtt{cT}_{prev}, \mathtt{P}_{prev})$-periodic; and (2) by showing that the period $\mathtt{P}$ and threshold $\mathtt{cT}$ are single-exponential in $n = |\mathcal{A}|$ and in the nesting depth of φ.

1. We show that for every state $s \in S$ and counters $v, v' > \mathtt{cT}$, if $v \equiv v' \mod \mathtt{P}$ then $(s, v) \models \varphi \iff (s, v') \models \varphi$. Withot loss of generality, write $v' = v + z \cdot \mathtt{P}$, for some $z \in \mathbb{N}$.
   - If $(s, v) \models \varphi$ then by the definition of the AU operator and the completeness of $\mathcal{A}$ we have i) there is a level ℓ, such that for every state e and counter value u, if $(s, v) \overset{\ell}{\rightsquigarrow} (e, u)$ then $(e, u) \models \psi_2$, and ii) for every level $m < \ell$, state h and counter value x, if $(s, v) \overset{m}{\rightsquigarrow} (h, x)$ then $(h, x) \models \psi_1$.

     Observe first that if $\ell \notin \mathtt{core}(v)$, then there also exists a level $\hat{\ell} < \ell$ witnessing $(s, v) \models \varphi$, such that $\hat{\ell} \in \mathtt{core}(v)$. Indeed, we obtain $\hat{\ell}$, by choosing the largest level $\hat{\ell}$ in the $\mathtt{core}$ of ℓ's segment, such that $\ell \equiv \hat{\ell} \mod \mathtt{P}$. As $\hat{\ell} < \ell$, it directly follows that for every level $\hat{m} < \hat{\ell}$, state $\hat{h}$ and counter value $\hat{x}$, if $(s, v) \overset{\hat{m}}{\rightsquigarrow} (\hat{h}, \hat{x})$ then $(\hat{h}, \hat{x}) \models \psi_1$. Now, assume toward contradiction that there is a state $\hat{e}$ and a counter value $\hat{u}$, such that $(s, v) \overset{\hat{\ell}}{\rightsquigarrow} (\hat{e}, \hat{u})$ and $(\hat{e}, \hat{u}) \not\models \psi_2$. Then by (possibly several applications of) Lemma 18.1b, there is also a counter value $\hat{\hat{u}} \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} \hat{u}$, such that $(s, v) \overset{\ell}{\rightsquigarrow} (\hat{e}, \hat{\hat{u}})$. As ψ_2 is $(\mathtt{cT}_{prev}, \mathtt{P}_{prev})$-periodic, we have $(\hat{e}, \hat{\hat{u}}) \not\models \psi_2$, leading to a contradiction.

     Next, we claim that the level $\ell' = \mathsf{shift}^z(\hat{\ell})$, obtained by z (recall that $z = \frac{v' - v}{\mathtt{P}}$) applications of the shift function on $\hat{\ell}$, witnesses $(s, v') \models \varphi$, namely that i) for every state e' and counter value u', if $(s, v') \overset{\ell'}{\rightsquigarrow} (e', u')$ then $(e', u') \models \psi_2$, and ii) for every level $m' < \ell'$, state h' and counter value x', if $(s, v') \overset{m'}{\rightsquigarrow} (h', x')$ then $(h', x') \models \psi_1$.

     Indeed, i) were it the case that $(e', u') \not\models \psi_2$ then by (z applications of) Lemma 18.2a, there was also a counter value $u'' \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} u'$, such that $(s, v) \overset{\hat{\ell}}{\rightsquigarrow} (e', u'')$ and therefore $(e', u'') \not\models \psi_2$, leading to a contradiction; and ii) were it the case that $(s, v') \overset{m'}{\rightsquigarrow} (h', x')$ and $(h', x') \not\models \psi_1$ then a) by Lemma 18.1a, as in the argument above, there is also a level $\tilde{m} \leq m'$, such that $\tilde{m}$ is in the core of m''s segment and $(s, v') \overset{\tilde{m}}{\rightsquigarrow} (h', \tilde{x})$ where $\tilde{x} \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} x$, and b) by Lemma 18.2a there is a level $\hat{m} < \hat{\ell}$, such that $(s, v) \overset{\hat{m}}{\rightsquigarrow} (h', \hat{x})$ where $\hat{x} \equiv_{\mathtt{cT}_{prev}, \mathtt{P}_{prev}} x$ and therefore $(h', \hat{x}) \not\models \psi_1$, leading to a contradiction.

- If $(s, v') \models \varphi$ then we have $(s, v) \models \varphi$ by an argument analogous to the above, while using Lemma 18.2b instead of Lemma 18.2a.

2. The threshold $\mathtt{cT}$ and period $\mathtt{P}$ are calculated along the induction on the structure of the formula φ. They start with threshold 0 and period 1, and their increase in each step of the induction depends on the outermost operator.
   Observe first that we can take as the worst case the same increase in every step, that of the UA case, since it guarantees the others. Namely, its required threshold, based on the threshold and period of the subformulas, is bigger than the threshold required in the other cases, and its required period is divisible by the periods required for the other cases.
   Next, notice that both the threshold and period in the UA case only depend on the periods of the subformulas (i.e., not on their thresholds), so it is enough to show that the period is singly exponential in n and the nesting depth of φ.
   The period in the UA case is defined to be $\mathtt{P}(\varphi) = B \cdot \mathrm{lcm}(\mathtt{P}(\psi_1), \mathtt{P}(\psi_2))$, where $B = \mathrm{lcm}[1..2b^3]$, and b is the bound on the length of a linear path scheme for $\mathcal{A}$. By [17], b is polynomial in n, and as $\mathrm{lcm}[1..2b^3] < 4^{2b^3}$, we get that B is singly exponential in n.
   Considering $\mathrm{lcm}(\mathtt{P}(\psi_1), \mathtt{P}(\psi_2)$, while in general $\mathrm{lcm}(x, y)$ of two numbers x and y might be equal to their multiplication, in our case, as both ψ_1 and ψ_2 are calculated along the induction via the same scheme above, they are both an exponent of B. Hence, $\mathrm{lcm}(\mathtt{P}(\psi_1), \mathtt{P}(\psi_2) = \max(\mathtt{P}(\psi_1), \mathtt{P}(\psi_2)$. Thus, we get that $\mathtt{P}(\varphi) \leq B^x$, where x is bounded by the nesting depth of φ. ◀

References

1. Shaull Almagor, Udi Boker, and Orna Kupferman. Formalizing and reasoning about quality. *Journal of the ACM*, 63(3):24:1–24:56, 2016.
2. Roland Axelsson, Matthew Hague, Stephan Kreutzer, Martin Lange, and Markus Latte. Extended computation tree logic. In *Logic for Programming, Artificial Intelligence, and Reasoning: 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings 17*, pages 67–81. Springer, 2010.
3. Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orana Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Log.*, 15(4):27:1–27:25, 2014.
4. Krishnendu Chatterjee and Laurent Doyen. Computation tree logic for synchronization properties. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 98:1–98:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.
5. Michael R Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K Micinski, Markus N Rabe, and César Sánchez. Temporal logics for hyperproperties. In *Principles of Security and Trust: Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings 3*, pages 265–284. Springer, 2014.
6. Michael R Clarkson and Fred B Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.
7. Byron Cook, Heidy Khlaaf, and Nir Piterman. Verifying increasingly expressive temporal logics for infinite-state systems. *Journal of the ACM (JACM)*, 64(2):1–39, 2017.
8. David C Cooper. Theorem proving in arithmetic without multiplication. *Machine intelligence*, 7(91-99):300, 1972.
9. Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Model-checking CTL* over flat Presburger counter systems. *Journal of Applied Non-Classical Logics*, 20(4):313–344, 2010.

10 Alain Finkel, Jérôme Leroux, and Grégoire Sutre. Reachability for two-counter machines with one test and one reset. In *FSTTCS 2018-38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 122, pages 31–1. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2018.

11 Michael J Fischer and Michael O Rabin. Super-exponential complexity of Presburger arithmetic. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 122–135. Springer, 1998.

12 Seymour Ginsburg and Edwin H Spanier. Bounded ALGOL-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.

13 Stefan Göller and Markus Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM J. Comput.*, 42(3):884–923, 2013.

14 Christoph Haase. A survival guide to Presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018.

15 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In *CONCUR 2009 – Concurrency Theory: 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings 20*, pages 369–383. Springer, 2009.

16 Jérôme Leroux and Grégoire Sutre. Flat counter automata almost everywhere! In *International Symposium on Automated Technology for Verification and Analysis*, pages 489–503. Springer, 2005.

17 Jérôme Leroux and Grégoire Sutre. Reachability in two-dimensional vector addition systems with states: One test is for free. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

18 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, N.J., 1967.

19 Mojzesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du I Congrès de Mathématiciens des Pays Slaves*, pages 92–101, 1929.

20 Leslie G Valiant and Michael S Paterson. Deterministic one-counter automata. *Journal of Computer and System Sciences*, 10(3):340–350, 1975.

21 Igor Walukiewicz. Model checking CTL properties of pushdown systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 127–138. Springer, 2000.

A Appendix – Omitted Proofs

A.1 Proofs for Sections 3 and 4

Proof of Lemma 6. In [17], the authors consider a model called 2-TVASS, comprising a 2-dimensional vector addition system where the first counter can also be tested for zero. In our setting, this corresponds to an OCA equipped with an additional counter that cannot be tested for zero. A configuration of a 2-TVASS is (s, x_1, x_2) describing the state and the values of the two counters.

We transform the OCA $\mathcal{A}$ to a 2-TVASS $\mathcal{A}'$ by introducing a length-counting component. That is, in every transition of $\mathcal{A}'$ as a 2-TVASS, the second component increments by 1. We thus have the following: there is a path of length ℓ from (s, v) to (s', v') in $\mathcal{A}$ if and only if there is a path from $(s, v, 0)$ to (s', v', ℓ) in $\mathcal{A}'$.

From [17, Corollary 16], using the fact that $\mathcal{A}'$ has only weights in $\{-1, 0, 1\}$, we have that if there is a path from (s, x_1, x_2) to (s', y_1, y_2) in $\mathcal{A}'$, then there is also such a π-shaped path where π is of flat length $\text{poly}(|S|)$ and size $O(|S|^3)$, which concludes the proof. ◀

Proof of Lemma 8. Consider the set $V_{\text{init}} = \{v \mid v \leq \mathtt{t}(\varphi) \text{ and } (s,v) \models \varphi\}$. We define

$$P_{\varphi,s}(x) := \bigvee_{u \in V_{\text{init}}} \exists m.\ x = u + m \cdot \mathtt{p}(\varphi)$$

The correctness of this formula is immediate from the definition of $V_{\text{init}}, \mathtt{t}(\varphi)$, and $\mathtt{p}(\varphi)$. In order to compute $P_{\varphi,s}(x)$, we need to compute V_{init}. This is done using Proposition 4, by evaluating φ from every state of the Kripke structure. ◀

Proof of Lemma 9. Recall from Observation 2 that the set of satisfying assignments for a PA formula in dimension 1 is effectively semilinear and can be written as $\text{Lin}(B, \{r\}) = \{b + \lambda r \mid b \in B,\ \lambda \in \mathbb{N}\}$ for effectively computable $B \subseteq \mathbb{N}$ and $r \in \mathbb{N}$.

For every state $s \in S$, consider therefore the set $\text{Lin}(B_s, \{r_s\})$ corresponding to $P_{\varphi,s}$. Define $\mathtt{t}(\varphi) = \max\{b \mid b \in B_s,\ s \in S\}$ and $\mathtt{p}(\varphi) = \text{lcm}(\{r_s\}_{s \in S}) \cdot \mathtt{t}(\varphi)$.

We claim that φ is $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$-periodic. Indeed, consider $v, v' > \mathtt{t}(\varphi)$ such that $v \equiv v' \mod \mathtt{p}(\varphi)$, and let $s \in S$. Without loss of generality assume $v > v'$, so we can write $v - v' = K \cdot \mathtt{p}(\varphi)$ for some $K \in \mathbb{N}$.

Now, if $(s, v') \models \varphi$, then $v' = b'_s + \lambda' r_s$ for some $b'_s \in B_s$ and $\lambda' \in \mathbb{N}$. Then, $v = b'_s + \lambda' r_s + K \cdot \mathtt{p}(\varphi) = b'_s + (\lambda' + K\frac{\mathtt{p}(\varphi)}{r_s})r_s$, and note that r_s divides $\mathtt{p}(\varphi)$, so we have $v \in \text{Lin}(B_s, \{r_s\})$ and therefore $(s, v) \models \varphi$.

Conversely, if $(s, v) \models \varphi$ then $v = b_s + \lambda r_s$ for some $b_s \in B_s$ and $\lambda \in \mathbb{N}$. Then, $v' = b_s + \lambda r_s - K \cdot \mathtt{p}(\varphi) = b_s + (\lambda - K\frac{\mathtt{p}(\varphi)}{r_s})r_s$. Since $v' > \mathtt{t}(\varphi) \geq b_s$, then $\lambda - K\frac{\mathtt{p}(\varphi)}{r_S} \geq 1$. Thus, $v' \in \text{Lin}(B_s, \{r_s\})$ so $(v', s) \models \varphi$. Hence, φ is $(\mathtt{t}(\varphi), \mathtt{p}(\varphi))$-periodic. ◀

Proof of Lemma 17. For $i = 0$, the first claim trivially holds, as it claims for a cycle with a slope at least $\mathtt{s}_1$, which holds for any cycle, and since $\frac{\mathtt{sT}}{2} > 2b^5 \cdot \mathtt{P}$, it must repeat at least $b^4 \cdot \mathtt{P}$ times. The second claim vacuously holds, as no path can reach $\mathtt{cT}_{prev}$ at Segment 0.

Consider $i \in [1..m]$. We prove each of the claims separately.

We show that there are at least $b^5 \cdot \mathtt{P}$ total cycle repetitions of the required slope, and since there are at most b different cycles, one of them should repeat at least $b^4 \cdot \mathtt{P}$ times.

1. Consider a basic path τ of length $\ell \geq \S_i(v) + \frac{\mathtt{sT}}{2}$, such that the counter values of τ stay above $\mathtt{cT}_{prev}$. Assume by way of contradiction that τ has less than $b^5 \cdot \mathtt{P}$ cycles with $\mathtt{s}_j$ for $j > i$. We will show that ℓ must fall short of $\S_i(v) + \frac{\mathtt{sT}}{2}$.
   Since there are at most $b^5 \cdot \mathtt{P} - 1$ cycles with $\mathtt{s}_j$ with $j > i$, each of length at most b, then the total length of τ spent in such cycles and in the simple paths of τ (of total length at most b) is $X \leq b(b^5 \cdot \mathtt{P} - 1) + b = b^6 \cdot \mathtt{P}$. The effect accumulated by the transitions in X is at most X, if all the relevant transitions are $+1$.
   The remaining part of the path, whose length is denoted by $Y = \ell - X$, is spent in cycles with $\mathtt{s}_j$ for $j \leq i$. Note that its effect is at most $(v + X - \mathtt{cT}_{prev})$, and therefore its length $Y \leq \frac{-1}{\mathtt{s}_i}(v + X - \mathtt{cT}_{prev})$.
   Observe that any actual path τ that satisfies the required constraints cannot be shorter than the above theoretical path in which there are X transitions of effect $(+1)$.
   Therefore, we have $\ell \leq X + Y \leq b^6 \cdot \mathtt{P} + \frac{-1}{\mathtt{s}_i}(v + b^6 \cdot \mathtt{P} - \mathtt{cT}_{prev}) = \frac{-1}{\mathtt{s}_i}(v - \mathtt{cT}_{prev}) + (\frac{-1}{\mathtt{s}_i} + 1)b^3 \cdot \mathtt{P} \leq \frac{-1}{\mathtt{s}_i}(v - \mathtt{cT}_{prev}) + (b^7 + b^6) \cdot \mathtt{P}$.
   Recall, however, that $\ell \geq \S_i(v) + \frac{\mathtt{sT}}{2} = (\frac{-1}{\mathtt{s}_i}(v - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P}) + (\frac{1}{2}b^9 \cdot \mathtt{P})$, leading to a contradiction, as $\frac{b^9}{2} > b^6 + b^7 + b^8$, for $b > 2$.
2. Consider a basic path τ of length $\ell < \S_{i+1}(v)$, such that the counter values of τ reach $\mathtt{cT}_{prev}$. Without loss of generality, we can assume that ℓ is the first time when τ reaches $\mathtt{cT}_{prev}$ (otherwise we will look at the prefix of τ that satisfies this). Assume by way of contradiction that τ has less than $b^5 \cdot \mathtt{P}$ cycles with $\mathtt{s}_j$ for $j < i$, we show that ℓ must fall after $\S_{i+1}(v)$.

Since there are at most $b^5 \cdot \mathtt{P} - 1$ cycles with $\mathtt{s}_j$ for $j \leq i$, each of length at most b, then the total length of τ spent in such cycles and in the simple paths of τ (of total length at most b) is $X \leq b(b^5 \cdot \mathtt{P} - 1) + b = b^6 \cdot \mathtt{P}$. The effect accumulated by the transitions in X is at least $-X$, if all the relevant transitions are -1.
The remaining part of the path, whose length is denoted by $Y = \ell - X$, is spent in cycles with $\mathtt{s}_j$ for $j > i$. Note that its effect is at least $(v - X - \mathtt{cT}_{prev})$, and therefore its length $Y \geq \frac{-1}{\mathtt{s}_{i+1}}(v - X - \mathtt{cT}_{prev})$.
Observe that any actual path τ that satisfies the required constraints cannot be shorter than the above theoretical path in which there are X transitions of effect (-1).
Therefore, we have $\ell \geq X + Y \geq b^6 \cdot \mathtt{P} + \frac{-1}{\mathtt{s}_{i+1}}(v - b^6 \cdot \mathtt{P} - \mathtt{cT}_{prev}) = \frac{-1}{\mathtt{s}_{i+1}}(v - \mathtt{cT}_{prev}) + (\frac{-1}{\mathtt{s}_{i+1}} - 1)b^6 \cdot \mathtt{P} \geq \frac{-1}{\mathtt{s}_{i+1}}(v - \mathtt{cT}_{prev}) - b^7 \cdot \mathtt{P}$.
Recall, however, that $\ell < \S_{i+1}(v) = \frac{-1}{\mathtt{s}_{i+1}}(v - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P}$, leading to a contradiction. ◀

A.2 Remaining Cases in the Proof of Lemma 18

We present remaining arguments for the proof.

A.2.1 Proof of Lemma 18.1b

Consider a length $\ell \notin \mathtt{core}(v)$, and let τ be a basic path of length $\ell - \mathtt{P}$ such that $(s, v) \overset{\ell - \mathtt{P}}{\rightsquigarrow} (e, u)$. We construct from τ a path τ' for $(s, v) \overset{\ell}{\rightsquigarrow} (e, u')$, such that $u \equiv u'$.

This part of the proof, showing that a path can be properly lengthened by $\mathtt{P}$, is partially analogous to part 1a of the proof, which shows that a path can be shortened by $\mathtt{P}$. We split the proof to the same cases as in part 1a, and whenever the proofs are similar enough, we refer to the corresponding case in part 1a.

Case 1b.1: The counter values in τ stay above $\mathtt{cT}_{prev}$

Since there is no position in τ with counter value $\mathtt{cT}_{prev}$, then in particular τ has no zero-transitions. We split to subcases.

- If there is a non-negative cycle c in τ, we can obtain τ' by adding $\frac{\mathtt{P}}{|c|}$ copies of c, having that the counter values in τ' are at least as high as the corresponding ones in τ. Observe that τ' is of length $\ell + \mathtt{P}$ from (s, v) to (e, u') with $u' = u + \mathit{effect}(c)\frac{\mathtt{P}}{|c|}$. Since we have $u' \geq u \geq \mathtt{cT}_{prev}$, then $u \equiv u'$.
- If there are in τ two cycles of different slopes, such that each of them repeats at least $b \cdot \mathtt{P}$ times, we can use Proposition 14 to properly lengthen τ into τ'.
- Otherwise, we are in the case that only a single cycle c with some negative slope $\mathtt{s}$ repeats at least $b \cdot \mathtt{P}$ times. Since $\ell - \mathtt{P} > \S_i(v) + \frac{\mathtt{sT}}{2}$, by Lemma 17.1, we have $\mathtt{s} > \mathtt{s}_i$. We can thus add $\frac{\mathtt{P}}{|c|}$ copies of c, guaranteed that the counter values in τ' remain above $\mathtt{cT}_{prev}$. Indeed, the maximal drop (i.e., the negation of the minimum cumulative counter effect in the path) D of τ is $X + Y$, where X stands for the repetitions of c, and Y for the rest. We have $X \leq -\mathtt{s} \cdot (\ell - \mathtt{P})$ (notice that $\mathtt{s}$ is negative) and $Y \leq b + (b-1)b^2\mathtt{P}$, considering the worst case in which all transitions out of cycles, as well as in the cycles that are not c are of effect (-1). Thus, $D \leq -\mathtt{s} \cdot (\ell - \mathtt{P}) + b^3\mathtt{P}$.
  As $\ell - \mathtt{P} < \ell < \S_{i+1}(v) = \frac{(v - \mathtt{cT}_{prev})}{-\mathtt{s}_{i+1}} - b^8 \cdot \mathtt{P}$, we have $D \leq \frac{\mathtt{s}}{\mathtt{s}_{i+1}}(v - \mathtt{cT}_{prev}) + b^3 \cdot \mathtt{P} - (-\mathtt{s})b^8 \cdot \mathtt{P}$.
  Since $\mathtt{s} > \mathtt{s}_i$, it follows that $\mathtt{s} \geq \mathtt{s}_{i+1}$, thus (as both $\mathtt{s}$ and $\mathtt{s}_{i+1}$ are negative), we have $D \leq (v - \mathtt{cT}_{prev}) + b^3 \cdot \mathtt{P} - (-\mathtt{s})b^8 \cdot \mathtt{P}$.

Recall that we aim to show that $v - D$ can "survive" a drop of up to $\mathtt{P}$ additional copies of c (i.e., at most $b \cdot \mathtt{P}$), while keeping the counter value above $\mathtt{cT}_{prev}$. Thus, we need to show that $v - ((v - \mathtt{cT}_{prev}) + b^3 \cdot \mathtt{P} - (-\mathtt{s})b^8\mathtt{P}) - b \cdot \mathtt{P} > \mathtt{cT}_{prev}$. It is left to show that $(-\mathtt{s})b^8 \cdot \mathtt{P} > (b^3 + b) \cdot \mathtt{P}$, which obviously holds, as $-\mathtt{s} \geq \frac{1}{b}$.

Case 1b.2: τ reaches counter value $\mathtt{cT}_{prev}$

Let $0 \leq z_f \leq z_u \leq \ell - \mathtt{P}$ be the first and ultimate positions in τ where the counter value is exactly $\mathtt{cT}_{prev}$. We split τ into three parts: $\tau_1 = \tau[0 \,..\, z_f), \tau_2 = \tau[z_f \,..\, z_u), \tau_3 = \tau[z_u \,..\, \ell - \mathtt{P}]$ as in the analogous case in 1a.

1. *The middle part $\tau_2 = \tau[z_f \,..\, z_u]$ or last part $\tau_3 = \tau[z_u \,..\, \ell - \mathtt{P}]$ are of length at least $b^8\mathtt{P}$.* These cases are analogous to their counterparts 1 and 2 of 1a.2, by adding cycle repetitions instead of removing them.
2. *Only the first part $\tau_1 = \tau[0 \,..\, z_f)$ of length at least $b^8\mathtt{P}$*

   If any of the other parts is long, we lengthen them. Otherwise, their combined length is less than $2b^8 \cdot \mathtt{P} < \frac{\mathtt{sT}}{2}$, implying that the first part τ_1 is longer than $\S_i(v) + \frac{\mathtt{sT}}{2}$.

   Hence, by Lemma 17, there are "fast" and "slow" cycles c_f and c_s, respectively, of slopes $\mathtt{s}_f < \mathtt{s}_s$, such that each of them repeats at least $b^4 \cdot \mathtt{P}$ times in τ_1.

   Thus, by Proposition 15, we can add and/or remove some repetitions of c_f and c_s, such that τ_1 is longer by exactly $\mathtt{P}$. Yet, we should ensure that τ' is valid, in the sense that its corresponding first part τ'_1 cannot get the counter value to 0. We show it by cases:

   - If c_f or c_s are balanced, we can lengthen without changing the remaining counters.
   - If there is a positive cycle c_+ that repeats at least $2b^2 \cdot \mathtt{P}$ times, then the counter value climbs by at least $2b^2 \cdot \mathtt{P}$ from its value v_x at position x where c_+ starts and the position y where its repetitions end. As the counter gets down to $\mathtt{cT}_{prev} < \mathtt{P}$ at the end of τ_1, and $\mathtt{cT}_{prev} < P$ (Proposition 16), there must be a negative cycle c_- that repeats at least $b \cdot \mathtt{P}$ times between position y and the first position after y that has the counter value $v_x + b^2 \cdot \mathtt{P}$. Hence, we can add repetitions of c_+ and c_-, as per Proposition 15, ensuring that the only affected values are above v_x.
   - Otherwise, both c_f and c_s are negative, implying that we add some repetitions of c_s and remove some repetitions of c_f. We further split into two subcases:
     - If c_f appears before c_s then there is no problem, as the only change of values will be their increase.
     - If c_s appears first, ending at some position x, while c_f starts at some position y, then a-priori it might be that repeating c_s up to $b \cdot \mathtt{P}$ more times, as per Proposition 15, will take the counter value to 0.
       Yet, observe that since there are at most $b - 2$ positive cycles, and each of them can repeat at most $2b^2 \cdot \mathtt{P} - 1$ times, the counter value v_x at position x, and along the way until position y, is at least $v_y - (b-2)2b^2 \cdot \mathtt{P}$, where v_y is the counter value at position y. As c_f repeats at least $b^4 \cdot \mathtt{P}$ times, we have $v_y \geq b^4 \cdot \mathtt{P}$. Thus $v_x \geq b^4 \cdot \mathtt{P} - 2(b-2)b^2 \cdot \mathtt{P} > b^2\mathtt{P}$. Hence, repeating c_s up to $b \cdot \mathtt{P}$ more times at position x cannot take the counter value to 0, until position y, as required.

Continuation of the Proof of Lemma 18.2a

We detail the subcases of Case 2a.2.2 that were not detailed in the main text:

- If there is such a relatively slow cycle c_s that ends in a position x before a position y in which the first really fast cycle c_f starts, then: If $\mathtt{s}_s = s_i$, we can simply remove $\frac{\mathtt{P}}{-\mathtt{s}_s \cdot |c_s|}$ repetitions of c_s, getting the desired counter and lengths changes.
  Otherwise, namely when $\mathtt{s}_s > s_i$, we claim that the counter value between positions x and y is high enough, allowing to remove some repetitions of c_s and c_f, as per Proposition 14. As c_s might be positive, removal of its repetitions might decrease the counter; the worst case is when all its transitions are of effect $(+1)$, and we shorten the path the most, which is bounded by $b \cdot \mathtt{P}$. To ensure that the counter remains above the value P after the removal and until position y, we thus need it to be at least $b \cdot \mathtt{P} + \mathtt{P}$ at all positions until y. Indeed, the path has only cycles of slope at least $\mathtt{s}_{i+1}$ until position $y < \S_i + \mathtt{sT}$. Therefore, its drop D is at most b, for the non-cyclic part, plus $(-\mathtt{s}_{i+1})(\S_i(v + \mathtt{P}) + \mathtt{sT})$ for the cyclic part. (Notice that $\mathtt{s}_{i+1}$ is negative.) Therefore, $D \leq b + (-\mathtt{s}_{i+1})(\S_i(v + \mathtt{P}) + \mathtt{sT})$. Taking the value of $\S_i(v + \mathtt{P})$, we have $D \leq b + (-\mathtt{s}_{i+1})(\frac{-1}{\mathtt{s}_i}(v + \mathtt{P} - \mathtt{cT}_{prev}) - b^8 \cdot \mathtt{P} + \mathtt{sT}) = b + \frac{\mathtt{s}_{i+1}}{\mathtt{s}_i}(v + \mathtt{P} - \mathtt{cT}_{prev}) + (\mathtt{s}_{i+1})(b^8 \cdot \mathtt{P} + \mathtt{sT})$. Since $\mathtt{s}_{i+1} \leq \frac{1}{b}$ and $\frac{\mathtt{s}_j}{\mathtt{s}_i} \leq 1 - \frac{1}{b^2}$, we have $D \leq b + (1 - \frac{1}{b^2})(v + \mathtt{P} - \mathtt{cT}_{prev}) + \frac{1}{b}(b^8 \cdot \mathtt{P} + \mathtt{sT}) \leq b + (1 - \frac{1}{b^2})(v + \mathtt{P} - \mathtt{cT}_{prev}) + (b^7 \cdot \mathtt{P} + b^8 \cdot \mathtt{P})$. Now, as the counter starts with value $v + \mathtt{P}$, we need to show that $E = v + \mathtt{P} - D - (b^2 + 1)\mathtt{P} \geq 0$. Indeed, $E \geq \frac{1}{b^2}(v + \mathtt{P}) - b^7 \cdot \mathtt{P} - b^8 \cdot \mathtt{P} - (b^2 + 1)\mathtt{P}$. As $v \geq \mathtt{cT} = b^{11} \cdot \mathtt{P}$, we have $E \geq (b^9 - b^7 - b^8 - b^2 - 1)\mathtt{P} > 0$, as required.
- Otherwise, namely when there is no relatively slow cycle c_s that repeats at least $b^4 \cdot \mathtt{P}$ times before the position y in which the first really fast cycle c_f starts:
  Let z be the position after y, in which the path has the smallest counter value v_z before a relatively slow cycle that repeats $b^4 \cdot \mathtt{P}$ times.
  If $v_z > \mathtt{P}$, we can remove some repetitions of the really fast and relatively slow cycles, as per Proposition 14; We only remove repetitions of the fast decreasing cycle, so as a result the counter will only grow, and if it is guaranteed to be above $\mathtt{P}$ when starting the path from counter value $v + \mathtt{P}$, it will not reach 0 when starting from counter value v.
  Otherwise, namely when $v_z \leq \mathtt{P}$, we can remove $\frac{\mathtt{P}}{-\mathtt{s}_f \cdot |c_f|}$ repetitions of the really fast cycle c_f, guaranteeing that τ_1' remains above 0 until position y.
  Observe, however, that the resulting path τ' will be longer than ℓ. Nevertheless, we claim that in this case, the portion of τ_1 from position z until its last position with counter value v_z can be further shorten without changing the path's effect. Indeed, there is a cycle that repeats at least $b^4 \cdot \mathtt{P}$ times in this part, having an absolute effect of at least $b^3 \cdot \mathtt{P}$, implying that this part of τ_1 climbs up to a value of at least $b^3 \cdot \mathtt{P}$, and down again, allowing for the removal of positive and negative cycles, as per Proposition 15, analogously to 1 in the proof of Lemma 18.1a.2.

A.2.2 Proof of Lemma 18.2b

We remark that Segment 0 is identical in the shortening and lengthening argument, and this is a repetition of the main text. For a path of length ℓ, we have in Segment 0 that $\mathsf{shift}(\ell) = \ell$, and indeed a path from v is valid from $v + \mathtt{P}$ and vise versa, as they do not hit $\mathtt{cT}_{prev}$: Their maximal drop is $\mathtt{sT}$, while $v \geq \mathtt{cT} > \mathtt{P} + \mathtt{sT} > \mathtt{cT}_{prev} + \mathtt{sT}$.

We turn to the ith segment, for $i \geq 1$. Consider a basic path τ for $(s, v) \overset{\ell}{\rightsquigarrow} (e, u)$. Recall that $\mathsf{shift}(\ell) = \ell + \frac{\mathtt{P}}{-\mathtt{s}_i} \in [\S_i(v + \mathtt{P}) \mathinner{..} \S_i(v + \mathtt{P}) + \mathtt{sT})]$. We construct from τ a path τ' for $(s, v + \mathtt{P}) \overset{\mathsf{shift}(\ell)}{\rightsquigarrow} (e, u')$, such that $u \equiv u'$, along the following cases.

Case 2b.1: The counter values in τ stay above $\mathtt{cT}_{prev}$

Since there is no position in τ with counter value $\mathtt{cT}_{prev}$, then in particular τ has no zero-transitions. Since $\ell > \S_{i-1}(v) + \frac{\mathtt{sT}}{2}$, by Lemma 17.1, there is a cycle c in τ of slope $\mathtt{s} > \mathtt{s}_{i-1}$, namely of slope $\mathtt{s} \geq \mathtt{s}_i$. We can thus get the required path τ', by adding $\frac{\mathtt{P}}{-\mathtt{s}\cdot|c|}$ repetitions of c. Indeed, the counter value of τ' at the position that c starts will be bigger by $\mathtt{P}$ than the counter value of τ at that position, while the counter of τ' after the repetitions of c will be at least as high as the counter value of τ at the corresponding position.

Case 2b.2: τ reaches counter value $\mathtt{cT}_{prev}$

Let $0 \leq z_f \leq z_u \leq \ell$ be the first and ultimate positions in τ where the counter value is exactly $\mathtt{cT}_{prev}$. We split τ into three parts: $\tau_1 = \tau[0 \,..\, z_f), \tau_2 = \tau[z_f \,..\, z_u), \tau_3 = \tau[z_u \,..\, \ell]$ (it could be that $z_f = z_u$, in which case the middle part is empty).

In order to accommodate with possible zero transitions, we should lengthen τ_1, such that the resulting first part τ_1' of the new path τ', which starts with counter value $v + \mathtt{P}$, will also end with counter value exactly $\mathtt{cT}_{prev}$. Since τ_1 reaches $\mathtt{cT}_{prev}$ and it is shorter than $\S_{i+1}(v)$, it has by Lemma 17.2 at least $b^4 \cdot \mathtt{P}$ repetitions of a "fast" cycle c_f of slope $\mathtt{s}_f \leq \mathtt{s}_i$.

1. If $\mathtt{s}_f = \mathtt{s}_i$ or τ_2 or τ_3 are of length at least $b^5 \cdot \mathtt{P}$, we can add $\frac{\mathtt{P}}{-\mathtt{s}_f\cdot|c_f|}$ repetitions of c_f in τ_1. Note that the resulting first part τ_1' of the new path τ' indeed ends with counter value $\mathtt{cT}_{prev}$. However, while when $\mathtt{s}_f = \mathtt{s}_i$ the resulting length of τ' will be $\mathsf{shift}(\ell)$, as required, in the case that $\mathtt{s}_f < \mathtt{s}_i$, we have that the resulting path τ' will be shorter than $\mathsf{shift}(\ell)$. Nevertheless, in this case, as τ_2 or τ_3 are of length at least $b^5 \cdot \mathtt{P}$, we can further lengthen τ_2 or τ_3 without changing their effect, as per Proposition 15, analogously to 1 or 2, respectively, in the proof of Lemma 18.1a.2.
2. Otherwise, we are in the case that τ_1 has a "really fast" cycle of slope $\mathtt{s}_f < \mathtt{s}_i$ that repeats at least $b^4 \cdot \mathtt{P}$ times, and both τ_2 or τ_3 are of length less than $b^5 \cdot \mathtt{P}$. We claim that in this case τ_1 must also have $b^4 \cdot \mathtt{P}$ repetitions of a "relatively slow" cycle c_s of slope $\mathtt{s}_s \geq \mathtt{s}_i$. The argument for that is the same as in 2 of part 2a of the proof. As the case of $\mathtt{s}_s = \mathtt{s}_i$ is handled in the previous subcase, we may further assume that $\mathtt{s}_s > \mathtt{s}_i$. We may thus add repetitions of c_f and c_s, as per Proposition 14, lengthening τ_1 by exactly $\mathsf{shift}(\ell) - \ell$, while also ensuring that it ends with counter value exactly $\mathtt{cT}_{prev}$.

A Class of Rational Trace Relations Closed Under Composition

Dietrich Kuske
Technische Universität Ilmenau, Germany

Abstract

Rational relations on words form a well-studied and often applied notion. While the definition in trace monoids is immediate, they have not been studied in this more general context. A possible reason is that they do not share the main useful properties of rational relations on words. To overcome this unfortunate limitation, this paper proposes a restricted class of rational relations, investigates its properties, and applies the findings to systems equipped with a pushdown that does not hold a word but a trace.

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

Keywords and phrases rational relations, Mazurkiewicz traces, preservation of rationality and recognizability

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.20

Acknowledgements I want to thank the anonymous reviewers whose knowledgeable comments led to a substantial improvement of this paper. This applies in particular to one deep review that pointed out the central role of the composition and its commutation with the natural homomorphism which resulted in clearer and more conceptional proofs.

1 Introduction

Rational relations form a classical and well-studied concept (cf. [15, 14, 4, 17]) that embraces homomorphisms, inverse homomorphisms as well as substitutions. Rational relations appear in the study of automatic structures [1, 18, 32], rational Kripke frames [3], graph databases [2], the representation of infinite graphs and automata [19, 27, 35, 28, 8, 7], and natural language processing [20]. One particular application of rational relations can be found in the theory of pushdown systems: the reachability relation is prefix recognizable [10, 16] and therefore a rational relation which implies that forwards and backwards reachability preserve the regularity of a set of configurations ([6] provides an alternative proof for the backwards reachability).

Also the second theme of this paper has a long and diverse research history starting with Cartier and Foata's work in combinatorics [9] and Mazurkiewicz's ideas about the semantics of concurrent systems [26] that he modelled as equivalence classes of words, called traces today. Much of the work in computer science has concentrated on recognizable sets of traces, on model checking and synthesis problems, and on combinatorics, see [11] for a comprehensive presentation of the theory of traces; many of these results have been extended to more general concurrent systems like concurrent automata (cf., e.g. [13]), message passing automata [25], and other abstract models of distributed automata (e.g. [12, 5]).

Recently, Köcher and the current author considered a generalization of pushdown systems where the stack's contents is not a word, but a trace [23]; these systems were called *cooperating multi-pushdown systems* or cPDS. Our main results state that the forwards reachability relation preserves the rationality and the backwards reachability the recognizability of sets of configurations; these proofs are fundamentally different since the former analyses and simplifies runs of the cPDS while the latter generalises the construction by Bouajjani et al. [6]. While the reachability relation of a classical pushdown system is prefix recognizable,

© Dietrich Kuske;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 20; pp. 20:1–20:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

we also observed that this is not the case for cPDS. In addition, Köcher [22] inferred from the main result that the reachability relation is a rational trace relation, but it was not clear whether this rationality could be used to prove the preservation results as in the word case.

In this paper, we first observe that rational trace relations do not compose which leads to the definition of *left-closed rational* (called *lc-rational* from now on) trace relations. The main body of this paper studies their properties, in particular Section 4 shows that the class of lc-rational trace relations is closed under composition, preserves rationality under right- and recognizability under left-application, and is closed under certain concatenations. In Section 5, these properties are used to show that the reachability relation of a cPDS is lc-rational. This allows a uniform proof of the preservation results from [23]. Section 6 returns to the theory of lc-rational relations and characterises them in the spirit of Nivat's theorem for rational word relations; this section also demonstrates that the lc-rationality of a rational relation is not even semi-decidable. In the final Section 7, we return to the consideration of rational trace relations: we characterise them as compositions of the inverse of an lc-rational relation with an lc-rational relation, we show that their left- and right-application transforms recognizable sets into rational ones, and finally provide the class of *closed-rational trace relations* that enjoys all nice properties of rational word relations: closure under composition, preservation of rationality and recognizability under left- and right-application, invertability, and a Nivat-style characterisation.

2 Preliminaries

A *dependence alphabet* is a pair (Σ, D) where Σ is some finite alphabet and $D \subseteq \Sigma \times \Sigma$ is a symmetric and irreflexive relation on Σ, called the *dependence relation. We fix a dependence alphabet* (Σ, D) *throughout this paper.* The *independence relation* I is the irreflexive part of its complement, i.e., $I = \{(a, b) \in \Sigma^2 \mid (a, b) \notin D, a \neq b\}$. For two words $u, v \in \Sigma^*$, we write $(u, v) \in I$ or $u\ I\ v$ if, $(a, b) \in I$ for any two letters a and b occurring in u and v, resp..

Let $\sim \subseteq \Sigma^* \times \Sigma^*$ denote the least congruence on the free monoid Σ^* satisfying $ab \sim ba$ for all $(a, b) \in I$. We therefore have $u \sim v$ iff u can be transformed into v by exchanging the order of consecutive independent letters. Then the quotient $\mathbb{M}(\Sigma, D) = \Sigma^*/{\sim}$ is a monoid with multiplication $[u] \cdot [v] = [uv]$ where $[u]$ denotes the equivalence class of the word u with respect to the congruence $\sim$. Usually, we denote the *trace monoid* $\mathbb{M}(\Sigma, D)$ by $\mathbb{M}$. The mapping $\eta\colon \Sigma^* \to \mathbb{M}\colon u \mapsto [u]$ is a monoid homomorphism (thus, $[u]$ and $\eta(u)$ are synonyms).

Let M be a monoid. A set $L \subseteq M$ is *rational* if it can be constructed from finite subsets of M using the operations union, multiplication, and Kleene star (i.e., generated submonoid). Rational subsets of a finitely generated free monoid are called *regular languages*; by Kleene's theorem, they coincide with the languages accepted by deterministic finite automata [21].

Note that also $M \times M$ is a monoid (with componentwise multiplication). Rational subsets of this monoid are called *rational relations on* M. For $M = \Sigma^*$, we will speak of rational word relations, for $M = \mathbb{M}$, of rational trace relations.

Let $\varphi\colon M \to N$ be a monoid homomorphism into the monoid N. Then we have the following (cf. [31, Prop. 6.2] and [34, Prop. II.1.17]):

(i) If $K \subseteq M$ is rational, then $\varphi(K) \subseteq N$ is rational.

(ii) If $L \subseteq N$ is rational and φ is surjective, then there exists a rational set $K \subseteq M$ with $\varphi(K) = L$.

In particular, assuming φ to be surjective, we have that $L \subseteq N$ is rational if, and only if, it is the φ-image of some rational subset of M. In this paper, we will need the following two special cases of this characterisation of rational subsets of N.

(i) The homomorphism $\eta\colon \Sigma^* \to \mathbb{M}$ is surjective. Hence a set $\mathcal{L} \subseteq \mathbb{M}$ of traces is rational if, and only if, there exists a regular language $L \subseteq \Sigma^*$ with $\eta(L) = \mathcal{L}$.

(ii) The mapping $\eta_2\colon \Sigma^* \times \Sigma^* \to \mathbb{M} \times \mathbb{M}\colon (u, v) \mapsto ([u], [v])$ is a surjective homomorphism. Hence a trace relation $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$ is rational if, and only if, there exists a rational word relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\eta_2(R) = \mathcal{R}$.
From now on, we will abuse notation and write $\eta(R)$ for $\eta_2(R)$.

Let, again, M be a monoid. A set $L \subseteq M$ is *recognizable* if there exists a homomorphism $\psi\colon M \to S$ into some finite monoid S such that $L = \psi^{-1}\psi(L)$.[1] If M is a finitely generated free monoid, then L is recognizable iff L is regular (cf. [34, Thm. II.2.1]).

Let $\varphi\colon M \to N$ be a surjective homomorphism into a monoid N, and $L \subseteq N$. Then L is recognizable if, and only if, $\varphi^{-1}(L) \subseteq M$ is recognizable (cf. [31, Prop. 6.3] for the implication "$\Rightarrow$" and [34, Cor. II.2.2 and II.2.12] for the equivalence). In this paper, we will need the following special case. The homomorphism $\eta\colon \Sigma^* \to \mathbb{M}$ is surjective. Therefore, a set $\mathcal{L} \subseteq \mathbb{M}$ of traces is recognizable if, and only if, the language $\eta^{-1}(\mathcal{L}) \subseteq \Sigma^*$ is regular.

Let M be a monoid, $L \subseteq M$ a subset of M, and $R \subseteq M \times M$ a binary relation on M. Then we set

$$L^R = \{y \in M \mid \exists x \in L\colon (x, y) \in R\} \text{ and } {}^RL = \{x \in M \mid \exists y \in L\colon (x, y) \in R\}\,.$$

If R is (the graph of) a function $f\colon M \to M$, then L^R is the image of L under f, i.e., $L^R = \{f(x) \mid x \in L\}$, and ${}^RL$ is the preimage of L under f, i.e., ${}^RL = \{x \in M \mid f(x) \in L\}$. Often, authors write $L\,R$ for L^R and $R\,L$ for ${}^RL$; I prefer this notation as it stresses the different roles played by the set L and the relation R.

The mapping $2^M \to 2^M\colon L \mapsto L^R$ is the *right-application of* R while the mapping $2^M \to 2^M\colon L \mapsto {}^RL$ is the *left-application of* R. Note that for $R^{-1} = \{(y, x) \mid (x, y) \in R\}$, we have ${}^RL = L^{R^{-1}}$ and, since $(R^{-1})^{-1} = R$, also ${}^{R^{-1}}L = L^R$.

For two sets $K, L \subseteq M$, let $K \leq_{\text{rat}} L$ if there exists a rational relation $R \subseteq M \times M$ such that $K = L^R$. Since the relation $\mathrm{Id}_{\Sigma^*} = \{(u, u) \mid u \in \Sigma^*\}$ is rational, the relation $\leq_{\text{rat}}$ is reflexive.

In this paper, we will regularly consider subsets of and binary relations on Σ^* and $\mathbb{M}$, resp. I hope to simplify understanding by using the following conventions:

- Subsets of Σ^* are denoted by plain capital letters K and L; binary relations on Σ^* are similarly denoted R, R_1, and R_2.
- Subsets of $\mathbb{M}$ are denoted by curly capital letters $\mathcal{K}$ and $\mathcal{L}$; binary relations on $\mathbb{M}$ are similarly denoted $\mathcal{R}$, $\mathcal{R}_1$, and $\mathcal{R}_2$.

3 Rational and left-closed word relations

We first recall four properties of rational word relations. These properties form the basis of many applications in proofs that use rational relations.

▶ **Theorem 3.1.** *Let $R, R_1, R_2 \subseteq \Sigma^* \times \Sigma^*$ be rational. Then*

(R1) *Left- and right-application of rational relations preserve regularity: if $L \subseteq \Sigma^*$ is regular, then L^R and ${}^RL$ are regular [4, Cor. III.4.2].*

[1] In [34], Sakarovitch gives another definition of "recognizability", but he also shows that his definition is equivalent to the one used in this paper [34, Prop. II.2.1].

(R2) *The composition of rational relations*

$$R_1 \circ R_2 := \{(u, w) \in \Sigma^* \times \Sigma^* \mid \exists v \in \Sigma^* \colon u \; R_1 \; v \; R_2 \; w\}$$

is rational [4, Thm. III.4.4]. It follows that the relation $\leq_{\mathrm{rat}}$ *is transitive.*

(R3) *The inverse of a rational relation* $R^{-1} := \{(v, u) \mid (u, v) \in R\}$ *is rational.*

(R4) *A relation* $R' \subseteq \Sigma^* \times \Sigma^*$ *is rational if, and only if, there exists an alphabet* Γ*, a regular language* $K \subseteq \Gamma^*$*, and homomorphisms* $f, g \colon \Gamma^* \to \Sigma^*$ *such that* $R' = \{(f(x), g(x)) \mid x \in K\}$ *[30], [4, Thm. III.3.2] ("Nivat's theorem").*

By the very definition, property (R3) holds for rational trace relations $\mathcal{R} \subseteq \mathbb{M}^2$. Also property (R4) holds for any monoid M in place of Σ^* and therefore in particular for relations on the trace monoid $\mathbb{M}$ [4, Prop. III.3.4].

The following example shows that (R2) fails for rational trace relations.

▶ **Example 3.2.** Suppose there are $a, b, c, d \in \Sigma$ with $(a, b) \in D$ and $(c, d) \in I$. Consider the rational word relations $R_1 = \{(a^m b^n, c^m d^n) \mid m, n \geq 1\} = \{(a, c)\}^+ \cdot \{(b, d)\}^+$ and $R_2 = \{(d^n c^m, b^n a^m) \mid m, n \geq 1\}$. Since $(c, d) \in I$, we get $d^n c^m \sim c^m d^n$ for any $m, n \geq 0$ and therefore

$$\begin{aligned}
\mathcal{R}_1 := \eta(R_1) &= \{([a^m b^n], [c^m d^n]) \mid m, n \geq 1\} \text{ and} \\
\mathcal{R}_2 := \eta(R_2) &= \{([d^n c^m], [b^n a^m]) \mid m, n \geq 1\} \\
&= \{([c^m d^n], [b^n a^m]) \mid m, n \geq 1\}\,.
\end{aligned}$$

Hence $\mathcal{R} := \mathcal{R}_1 \circ \mathcal{R}_2 = \{([a^m b^n], [b^n a^m]) \mid m, n \geq 1\}$. Suppose the trace relation $\mathcal{R}$ is rational. Then there exists a rational word relation R with $\eta(R) = \mathcal{R}$. Since $(a, b) \in D$, $a^m b^n$ is the only word w with $\eta(w) = [a^m b^n]$ (similarly, $[b^n a^m] = \{b^n a^m\}$). Hence $\eta(R) = \mathcal{R}$ implies $R = \{(a^m b^n, b^n a^m) \mid m, n \geq 1\}$. But this relation is not rational which can easily be shown using Theorem 3.1 and pumping in the language K provided there. Hence the composition $\mathcal{R}$ of the two rational trace relations $\mathcal{R}_1$ and $\mathcal{R}_2$ is not rational.

▶ Remark. In a finitely generated free monoid Σ^*, a set is rational if, and only if, it is recognizable. In the trace monoid, any recognizable set is rational, but the converse implication does not hold, i.e., rationality and recognizability do not coincide (e.g., $\mathcal{L} = \{[ab]\}^*$ with $(a, b) \in I$ is rational, but not recognizable since $\eta^{-1}(\mathcal{L}) = \{u \in \{a, b\}^* \colon |u|_a = |u|_b\}$). Hence, there are two possible versions of (R1) for the trace monoid; later, we will see that none of them holds (Lemma 4.9).

We aim at a class $\mathbb{C}_{\mathbb{M}}$ of rational trace relations that is closed under composition. Recall that a trace relation $\mathcal{R}$ is rational if and only if there exists a rational word relation R with $\mathcal{R} = \eta(R)$. We will therefore first define a class $\mathbb{C}_{\Sigma^*}$ of rational word relations R that is closed under composition and satisfies

$$\eta(R_1 \circ R_2) = \eta(R_1) \circ \eta(R_2) \tag{1}$$

for any relations R_1 and R_2 in $\mathbb{C}_{\Sigma^*}$ (setting $\mathbb{C}_{\mathbb{M}} = \{\eta(R) \mid R \in \mathbb{C}_{\Sigma^*}\}$ will then ensure that $\mathbb{C}_{\mathbb{M}}$ is closed under composition, cf. Definition 4.1).

Note that Eq. (1) fails in Example 3.2 since there, $R_1 \circ R_2 = \emptyset$ and $\mathcal{R}_1 \circ \mathcal{R}_2 \neq \emptyset$. The reason is that there are words $t, u, u', v' \in \Sigma^*$ such that $(t, u) \in R_1$, $(u', v') \in R_2$, and $u \sim u'$ distinct such that $([t], [v']) \in \mathcal{R}_1 \circ \mathcal{R}_2$, but $(t, v') \notin R_1 \circ R_2$. The following definition circumvents this problem.

▶ **Definition 3.3.** *A relation $R \subseteq \Sigma^* \times \Sigma^*$ is* left-closed *if $\sim \circ R \subseteq R \circ \sim$, i.e.,*

$$\big(\exists u' \in \Sigma^* \colon u \sim u' \; R \; v'\big) \implies \big(\exists v \in \Sigma^* \colon u \; R \; v \sim v'\big)$$

holds for all $u, v \in \Sigma^*$. The relation R is lc-rational *if it is left-closed and rational.*

A very simple example is the identity relation $\mathrm{Id}_{\Sigma^*} = \{(u,u) \mid u \in \Sigma^*\}$: if $u \sim u' \; \mathrm{Id}_{\Sigma^*} \; v'$, then $u' = v'$. Setting $v = u$, we obtain $u \; \mathrm{Id}_{\Sigma^*} \; v = u \sim u' = v'$. Since this relation is clearly rational, it is indeed lc-rational. Other examples are $\Sigma^* \times \{\varepsilon\}$ and $\{\varepsilon\} \times \Sigma^*$.

▶ **Example 3.4.** Another, more demanding example, is the superword-relation: $u \in \Sigma^*$ is a *superword* of $v \in \Sigma^*$ if $u = u_1 v_1 u_2 v_2 \cdots u_n v_n u_{n+1}$ and $v = v_1 v_2 \cdots v_n$ for some $n \in \mathbb{N}$ and $u_1, u_2, \ldots, u_{n+1}, v_1, v_2, \ldots, v_n \in \Sigma^*$. In this case, we write $u \succeq v$. The superword-relation is rational since $\succeq = \{(a,a), (a,\varepsilon) \mid a \in \Sigma\}^*$.

Now suppose $u = x\,ab\,y$ with $(a,b) \in I$ and $x\,ba\,y = u' \succeq v'$. If $v' = \varepsilon$, we get $u \succeq \varepsilon \sim v'$. So suppose $v' \neq \varepsilon$. Then there are $n \in \mathbb{N}$, $u_1 \in \Sigma^*$, $u_2, \ldots, u_n, v_1, v_2, \ldots, v_n \in \Sigma^+$, and $u_{n+1} \in \Sigma^*$ such that $v_1 v_2 \cdots v_n = v'$ and $u_1 v_1 u_2 v_2 \cdots u_n v_n u_{n+1} = u' = x\,ab\,y$. This gives two decompositions of the word u' into the blocks u_i and v_i on the one hand, and into the factors x, ba, and y on the other hand. The factor ba from the second factorization can be covered by some factor u_i, by some factor v_i, or it belongs to some consecutive factors u_i and v_i or v_i and u_{i+1} of the former factorisation – in any of these cases, we construct a word v with $u \succeq v \sim v'$ (see appendix). Since $\sim$ is the least equivalence relation identifying $xaby$ with $xbay$ for $(a,b) \in I$, this proves that the superword-relation $\succeq$ is left-closed.

The subword-relation $\preceq$ is the inverse of the superword-relation. Suppose $(a,c) \in I$ and $(a,b), (b,c) \in D$. Then $ca \sim ac \preceq abc$, but there is no superword of ca that is equivalent to abc. Hence the subword-relation is not left-closed.

We next show that the class of lc-rational word relations has the properties desired: it is closed under composition and the homomorphism η commutes with composition.

▶ **Proposition 3.5.** *Let $R_1, R_2 \subseteq \Sigma^* \times \Sigma^*$.*

(i) *If R_2 is left-closed, then Eq. (1) holds, i.e., $\eta(R_1 \circ R_2) = \eta(R_1) \circ \eta(R_2)$.*

(ii) *If R_1 and R_2 are lc-rational, then also $R_1 \circ R_2$ is lc-rational.*

Proof. To demonstrate the first claim, let R_2 be lc-rational. For the inclusion $\eta(R_1 \circ R_2) \subseteq \eta(R_1) \circ \eta(R_2)$, let $(u,w) \in R_1 \circ R_2$. Then there exists $v \in \Sigma^*$ with $u \; R_1 \; v \; R_2 \; w$ and therefore $[u] \; \eta(R_1) \; [v] \; \eta(R_2) \; [w]$ implying $([u],[w]) \in \eta(R_1) \circ \eta(R_2)$.

For the converse inclusion, let $(x,z) \in \eta(R_1) \circ \eta(R_2)$. There is some trace y with $x \; \eta(R_1) \; y \; \eta(R_2) \; z$. Hence there are words u, v, v', w with

- $x = [u]$, $y = [v]$, and $(u,v) \in R_1$ and
- $y = [v']$, $z = [w]$, and $(v',w) \in R_2$.

Hence we have $u \; R_1 \; v \sim v' \; R_2 \; w$. Since R_2 is left-closed, there exists a word $w' \in \Sigma^*$ such that $u \; R_1 \; v \; R_2 \; w' \sim w$. Hence we have $(x,z) = ([u],[w]) = ([u],[w']) \in \eta(R_1 \circ R_2)$. This finishes the verification of the first claim.

Now, assume both relations R_1 and R_2 to be left-closed such that $\sim \circ R_i \subseteq R_i \circ \sim$ holds for all $i \in [2]$. Consequently, we get $\sim \circ R_1 \circ R_2 \subseteq R_1 \circ \sim \circ R_2 \subseteq R_1 \circ R_2 \circ \sim$. Hence, indeed, $R_1 \circ R_2$ is left-closed such that the second claim follows using Theorem 3.1(R2). ◀

The following proposition characterises the lc-rational word relations of the form $K \times L$ for languages $K, L \subseteq \Sigma^*$. This characterisation should also explain the name "left-closed".

▶ **Proposition 3.6.** *Let $K, L \subseteq \Sigma^*$ be non-empty.*

(i) *$K \times L$ is rational iff K and L both are regular.*

(ii) *$K \times L$ is left-closed iff K is* closed*, i.e., $u \sim u' \in K$ ensures $u \in K$.*

Consequently, $K \times L$ is lc-rational if, and only if, K and L are regular and K is closed.

Proof. The first claim is well-known, we present a proof in the appendix.

Suppose $K \times L$ to be left-closed and let $u \sim u' \in K$. Since L is non-empty, there exists $v' \in L$ implying $u \sim u'\ (K \times L)\ v'$. Since $K \times L$ is assumed to be left-closed, there exists $v \in \Sigma^*$ with $u\ (K \times L)\ v \sim v'$. This implies in particular $u \in K$. Hence, K is closed.

Conversely, suppose K to be closed and let $u \sim u'\ (K \times L)\ v'$. Then $u \sim u' \in K$ implying $u \in K$ such that (with $v = v'$) we get $u\ (K \times L)\ v \sim v'$, i.e., $K \times L$ is left-closed. ◀

4 The theory of lc-rational trace relations I

4.1 Definition, examples, composition

Recall that a trace relation $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$ is rational iff there exists a rational word relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\eta(R) = \mathcal{R}$. Similarly, we now lift the concept of lc-rational relations from words to traces.

▶ **Definition 4.1.** *A relation $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$ is* lc-rational *if there exists some lc-rational word relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\mathcal{R} = \eta(R)$.*

Simple examples are $\mathbb{M} \times \{[\varepsilon]\}$ and $\{[\varepsilon]\} \times \mathbb{M}$ since $\Sigma^* \times \{\varepsilon\}$ and $\{\varepsilon\} \times \Sigma^*$ are lc-rational word relations.

▶ **Example 4.2.** Another, more involved example, is the supertrace-relation [24]: $x \in \mathbb{M}$ is a *supertrace* of $y \in \mathbb{M}$ if $x = x_1 y_1 \cdots x_n y_n x_{n+1}$ and $y = y_1 y_2 \cdots y_n$ for some $n \in \mathbb{N}$ and $x_1, x_2, \ldots, x_{n+1}, y_1, y_2, \ldots, y_n \in \mathbb{M}$. In this case, we write $x \sqsupseteq y$. It is easily checked that $x \sqsupseteq y$ if, and only if, there are words u and v such that $x = [u]$, $y = [v]$, and $u \succeq v$, i.e., $\sqsupseteq\ = \eta(\succeq)$. Since the superword-relation $\succeq$ is lc-rational by Example 3.4, we obtain that the supertrace-relation $\sqsupseteq$ is lc-rational.

Also the identity relation $\mathrm{Id}_{\mathbb{M}} = \{(x, x) \mid x \in \mathbb{M}\}$ is lc-rational since Id_{Σ^*} is an lc-rational word relation. Note that $\mathrm{Id}_{\mathbb{M}}$ can be seen as a homomorphism from $\mathbb{M}$ to $\mathbb{M}$. We now generalize this example and show that every homomorphism is an lc-rational relation.

▶ **Lemma 4.3.** *Let $\varphi\colon \mathbb{M} \to \mathbb{M}$ be a homomorphism. Then $\mathcal{R} = \{(x, \varphi(x)) \mid x \in \mathbb{M}\}$ is lc-rational.*

Proof. Choose, for each $a \in \Sigma$, a word $w_a \in \Sigma^*$ with $\varphi([a]) = [w_a]$. Let $h\colon \Sigma^* \to \Sigma^*$ be the homomorphism with $h(a) = w_a$ for all $a \in \Sigma^*$. Then $\varphi([v]) = [h(v)]$ for all words $v \in \Sigma^*$.

Now consider h as a relation, i.e., set $R = \{(u, h(u)) \mid u \in \Sigma^*\}$. Then $\eta(R) = \mathcal{R}$ and $R = \{(a, w_a) \mid a \in \Sigma\}^*$ implying that the relation R is rational.

We next verify that R is left-closed. So let $u, u', v' \in \Sigma^*$ with $u \sim u'\ R\ v'$. From $u'\ R\ v'$, we obtain $v' = h(u')$. With $v := h(u)$, we obtain

$$[v'] = \varphi([u']) = \varphi([u]) = [h(u)] = [v]$$

implying $v' \sim v$. Hence we have $u\ R\ v \sim v'$ implying that R is left-closed. ◀

Later, we will characterize the lc-rational relations among the direct products $\mathcal{K} \times \mathcal{L}$ of sets of traces $\mathcal{K}$ and $\mathcal{L}$ (we have done so for word relations in Proposition 3.6). But first, we do this for the special case that one of the sets $\mathcal{K}$ and $\mathcal{L}$ equals $\{[\varepsilon]\}$.

▶ **Lemma 4.4.** *Let $\mathcal{K} \subseteq \mathbb{M}$ be a set of trace.*

(i) *$\mathcal{K} \times \{[\varepsilon]\}$ is lc-rational if, and only if, $\mathcal{K}$ is recognizable.*

(ii) *$\{[\varepsilon]\} \times \mathcal{K}$ is lc-rational if, and only if, $\mathcal{K}$ is rational*

Proof. First, suppose $\mathcal{K} \times \{[\varepsilon]\}$ is lc-rational. Then there exists an lc-rational word relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\eta(R) = \mathcal{K} \times \{[\varepsilon]\}$. Since ε is the only word w with $\eta(w) = [\varepsilon]$, there exists a language $K \subseteq \Sigma^*$ with $R = K \times \{\varepsilon\}$. Now Proposition 3.6 implies that K is closed and regular. From $\eta(K \times \{\varepsilon\}) = \eta(R) = \mathcal{K} \times \{[\varepsilon]\}$, we obtain $\eta(K) = \mathcal{K}$ and therefore $K \subseteq \eta^{-1}(\mathcal{K})$. For the converse inclusion, let $u \in \eta^{-1}(\mathcal{K})$ implying $[u] \in \mathcal{K}$. From $\eta(K) = \mathcal{K}$, we obtain some word $u' \in K$ with $[u'] = [u]$ and therefore $u \sim u' \in K$. Since K is closed, this implies $u \in K$. Hence, we showed $K = \eta^{-1}(\mathcal{K})$. Since K is regular, the set $\mathcal{K}$ is indeed recognizable.

Next, suppose $\mathcal{K}$ is recognizable. Then $K := \eta^{-1}(\mathcal{K})$ is regular. It is easily seen that K is closed such that, again by Proposition 3.6, the relation $R := K \times \{\varepsilon\}$ is lc-rational. But $\eta(R) = \mathcal{K} \times \{[\varepsilon]\}$, hence this direct product is lc-rational as well. This completes the proof of the first claim.

The second claim is shown similarly, one need not show $K = \eta^{-1}(\mathcal{K})$, but only $\eta(K) = \mathcal{K}$ as we have done above. ◀

By Example 3.2, the composition of rational trace relations is, in general, not rational. The following lemma demonstrates that the composition is rational provided the second relation is lc-rational. If both relations are lc-rational, then the composition is so as well.

▶ **Proposition 4.5.** *Let $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathbb{M}^2$ be rational trace relations.*

(i) *If $\mathcal{R}_2$ is lc-rational, then $\mathcal{R}_1 \circ \mathcal{R}_2$ is rational.*

(ii) *If $\mathcal{R}_1$ and $\mathcal{R}_2$ both are lc-rational, then $\mathcal{R}_1 \circ \mathcal{R}_2$ is even lc-rational.*

Proof. There exists a rational word relation $R_1 \subseteq \Sigma^* \times \Sigma^*$ with $\eta(R_1) = \mathcal{R}_1$, and there exists an lc-rational word relation $R_2 \subseteq \Sigma^* \times \Sigma^*$ with $\eta(R_2) = \mathcal{R}_2$. By Theorem 3.1, the composition $R_1 \circ R_2$ is rational and satisfies $\eta(R_1 \circ R_2) = \eta(R_1) \circ \eta(R_2)$ by Proposition 3.5(i). Hence $\mathcal{R}_1 \circ \mathcal{R}_2$ is the η-image of the rational word relation $R_1 \circ R_2$ and therefore rational.

If, in addition, $\mathcal{R}_1$ is also lc-rational, then the relation R_1 can be assumed to be lc-rational. Hence, by Proposition 3.5(ii), the composition $R_1 \circ R_2$ is even lc-rational. Hence $\mathcal{R}_1 \circ \mathcal{R}_2$ is the η-image of lc-rational word relation $R_1 \circ R_2$ and therefore lc-rational. ◀

Proposition 3.6 characterises the lc-rational word relations of the form $K \times L$. We can now lift this result to the trace setting.

▶ **Proposition 4.6.** *Let $\mathcal{K}, \mathcal{L} \subseteq \mathbb{M}$ be non-empty.*

(i) *$\mathcal{K} \times \mathcal{L}$ is rational iff $\mathcal{K}$ and $\mathcal{L}$ both are rational.*

(ii) *$\mathcal{K} \times \mathcal{L}$ is lc-rational iff $\mathcal{K}$ is recognizable and $\mathcal{L}$ is rational.*

Proof. The first claim can be shown as the first claim of Proposition 3.6 (see appendix).

Next, let $\mathcal{K}$ be recognizable and $\mathcal{L}$ rational. Then the trace relations $\mathcal{K} \times \{[\varepsilon]\}$ and $\{[\varepsilon]\} \times \mathcal{L}$ are lc-rational by Lemma 4.4. Note that

$$\mathcal{K} \times \mathcal{L} = (\mathcal{K} \times \{[\varepsilon]\}) \circ (\{[\varepsilon]\} \times \mathcal{L}) \,.$$

Hence the relation $\mathcal{K} \times \mathcal{L}$ is the composition of two lc-rational relations. Now Proposition 4.5(ii) implies that $\mathcal{K} \times \mathcal{L}$ is lc-rational as well.

Conversely, suppose $\mathcal{K} \times \mathcal{L}$ to be lc-rational. Since $\mathbb{M}$ is recognizable, the relation $\mathbb{M} \times \{[\varepsilon]\}$ is lc-rational by Lemma 4.4. Since

$$(\mathcal{K} \times \mathcal{L}) \circ (\mathbb{M} \times \{[\varepsilon]\}) = \mathcal{K} \times \{[\varepsilon]\}),$$

the relation $\mathcal{K} \times \{[\varepsilon]\}$ is lc-rational by Proposition 4.5. Now $\mathcal{K}$ is recognizable by Lemma 4.4(i).

For the rationality of $\mathcal{L}$, we can argue similarly (using that $\mathbb{M}$ is rational this time). ◀

4.2 Preservation of language properties

Recall that rational word relations preserve the regularity of languages under left- and right-application. Since rationality and recognizability are different notions in the trace monoid, this leads to two possible generalisations; later, Lemma 4.9 below will show that none of them holds for rational trace relations.

Now restrict attention to lc-rational trace relations $\mathcal{R}$. Since $\mathcal{R}^{-1}$ need not be lc-rational, we now get four possible preservation results: we could consider rationality or recognizability as well as left- or right-application. The following lemma shows that two of them hold, Lemma 4.9 proves that the other two fail.

▶ **Theorem 4.7.** *Let $\mathcal{R} \subseteq \mathbb{M}^2$ be an lc-rational trace relation.*

(i) *If $\mathcal{L} \subseteq \mathbb{M}$ is recognizable, then also ${}^{\mathcal{R}}\mathcal{L}$ is recognizable.*

(ii) *If $\mathcal{L} \subseteq \mathbb{M}$ is rational, then also $\mathcal{L}^{\mathcal{R}}$ is rational.*

Proof. Let $\mathcal{L}_1$ be recognizable and $\mathcal{L}_2$ rational. Then $\mathcal{L}_1 \times \{[\varepsilon]\}$ and $\{[\varepsilon]\} \times \mathcal{L}_2$ are lc-rational by Lemma 4.4 and we have

$${}^{\mathcal{R}}\mathcal{L}_1 \times \{[\varepsilon]\} = \mathcal{R} \circ (\mathcal{L}_1 \times \{[\varepsilon]\}) \text{ and } \{[\varepsilon]\} \times \mathcal{L}_2^{\mathcal{R}} = (\{[\varepsilon]\} \times \mathcal{L}_2) \circ \mathcal{R}.$$

Hence ${}^{\mathcal{R}}\mathcal{L}_1 \times \{[\varepsilon]\}$ and $\{[\varepsilon]\} \times \mathcal{L}_2^{\mathcal{R}}$ are lc-rational relations by Proposition 4.5. Using Lemma 4.4 again, we obtain that ${}^{\mathcal{R}}\mathcal{L}_1$ is recognizable and $\mathcal{L}_2^{\mathcal{R}}$ is rational. ◀

Define the relation $\leq_{\mathrm{lc-rat}}$ in the same way as the relation $\leq_{\mathrm{rat}}$, but with the restriction to lc-rational relations: $\mathcal{K} \leq_{\mathrm{lc-rat}} \mathcal{L}$ if there exists an lc-rational relation $\mathcal{R}$ such that $\mathcal{K} = \mathcal{L}^{\mathcal{R}}$. Then Proposition 4.5 trivially implies the following.

▶ **Corollary 4.8.** *The relation $\leq_{\mathrm{lc-rat}}$ is transitive.*

We now come to the announced non-preservation results; they hold for lc-rational trace relations and therefore, in particular, for the larger class of rational trace relations.

▶ **Lemma 4.9.** *Suppose there are $a, b, c, d \in \Sigma$ with $(a, b) \in D$ and $(c, d) \in I$.*

There exists an lc-rational relation $\mathcal{R} \subseteq \mathbb{M}^2$, a recognizable set $\mathcal{K} \subseteq \mathbb{M}$, and a rational set $\mathcal{L} \subseteq \mathbb{M}$ such that ${}^{\mathcal{R}}\mathcal{K}$ is not rational and $\mathcal{L}^{\mathcal{R}}$ is not recognizable.

Proof. Let $R = \{(a, c), (b, d)\}^*$. Then R is rational and, since $(a, b) \in D$, even lc-rational. We consider the lc-rational trace relation $\mathcal{R} = \eta(R)$. Since $(c, d) \in I$, we obtain $([u], [v]) \in \mathcal{R}$ if, and only if, $u \in \{a, b\}^*$, $v \in \{c, d\}^*$, $|u|_a = |v|_c$, and $|u|_b = |v|_d$.

Consider the regular language $K = \{cd\}^*$ and let $\mathcal{K}$ denote the rational set $\eta(K)$. Since $(c, d) \in I$, we get $[v] \in \mathcal{K}$ iff $v \in \{c, d\}^*$ and $|v|_c = |v|_d$. It follows that $[u] \in {}^{\mathcal{R}}\mathcal{K}$ iff $u \in \{a, b\}^*$ and $|u|_a = |u|_b$. Let $H \subseteq \Sigma^*$ denote the set of words $u \in \{a, b\}^*$ with $|u|_a = |u|_b$. Since $(a, b) \in D$, this language H is the only language with $\eta(H) = {}^{\mathcal{R}}\mathcal{K}$. Since H is not regular, it follows that ${}^{\mathcal{R}}\mathcal{K}$ is not rational which proves the first claim.

Next, let $\mathcal{L} = \{[ab]\}^*$. Then $[u] \in \mathcal{L}$ iff $u \in \{ab\}^*$ since $(a, b) \in D$. Hence $\eta^{-1}(\mathcal{L})$ is the regular language $L = \{ab\}^*$, i.e., $\mathcal{L}$ is recognizable. Note that $\mathcal{L}^{\mathcal{R}}$ is the set of traces $[v]$ with $v \in \{c, d\}^*$ and $|v|_c = |v|_d$ (i.e., it equals $\mathcal{K}$). Hence the preimage of $\mathcal{L}^{\mathcal{R}}$ under η is not regular, i.e., $\mathcal{L}^{\mathcal{R}}$ is not recognizable. ◀

4.3 Products of lc-rational relations

By the very definition, the (componentwise) product of two rational relations is rational again: if $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathbb{M}^2$ are rational, then so is the relation

$$\mathcal{R}_1 \cdot \mathcal{R}_2 = \{(xx', yy') \mid (x, y) \in \mathcal{R}_1, (x', y') \in \mathcal{R}_2\}\,.$$

Lemma A.1 in the appendix demonstrates that this does not hold for lc-rational relations. We now come to two special cases of relations $\mathcal{R}_1$ that ensure the lc-rationality of $\mathcal{R}_1 \cdot \mathcal{R}_2$:

▶ **Lemma 4.10.** *Let $\mathcal{K} \subseteq \mathbb{M}$ be recognizable. Then the relation $\mathcal{R} = (\mathcal{K} \times \{[\varepsilon]\}) \cdot \mathrm{Id}_{\mathbb{M}} = \{(xy, y) \mid x \in \mathcal{K}, y \in \mathbb{M}\}$ is lc-rational.*

Proof. The language $K := \eta^{-1}(\mathcal{K})$ is regular. Let R denote the set of pairs of words

$$(u_1 v_1 u_2 v_2 \cdots u_n v_n, v_1 v_2 \cdots v_n)$$

with $n \in \mathbb{N}$ and $u_1, u_2, \ldots, u_n, v_1, \ldots, v_n \in \Sigma^*$ such that

(i) $u_1 u_2 \cdots u_n \in K$,
(ii) $v_1 \cdots v_n \in \Sigma^*$, and
(iii) $(v_1 v_2 \cdots v_i, u_{i+1}) \in I$ for all $i \in [n-1]$.

We will show that R is rational, that $\eta(R) = \mathcal{R}$, and that R is left-closed.

To verify that R is a rational relation, we present a rational transducer $\mathcal{T}$ which is a non-deterministic finite automaton with edges labeled by elements from $\Sigma^* \times \Sigma^*$ (cf. [4, p. 77]). Since K is regular, there is a non-deterministic finite automaton $\mathcal{A} = (Q, \Sigma, Q_0, T, F)$ (where Q_0 and F are the initial and accepting states and $T \subseteq Q \times \Sigma \times Q$) accepting K. States of the transducer $\mathcal{T}$ are pairs (q, A) of a state $q \in Q$ and a set of letters $A \subseteq \Sigma$. A state (q, A) is initial if $q \in Q_0$ and $A = \emptyset$ and it is accepting if $q \in F$. The transducer has two types of transitions (with $a \in \Sigma$):

- There is a transition from (p, A) to (q, B) labeled (a, a) iff $p = q$ and $B = A \cup \{a\}$.
- There is a transition from (p, A) to (q, B) labeled (a, ε) iff $(p, a, q) \in T$ is a transition of the automaton $\mathcal{A}$, $\{a\} \times A \subseteq I$, and $A = B$.

Let $p \in Q_0$, $q \in Q$, $A \subseteq \Sigma$, and $v, w \in \Sigma^*$. Then the transducer $\mathcal{T}$ has a path labeled (w, v) from $(p, \emptyset)$ to (q, A) iff the following hold:

- A is the set of letters of v.
- w results from v by injecting some letters (using transitions labeled (a, ε)) that are independent from all letters of v read so far.
- The sequence u of injected letters leads from p to q in the automaton $\mathcal{A}$.

Consequently, (w, v) labels a path from some initial to some accepting state iff $(w, v) \in R$. Hence, indeed, R is rational [4, Thm. III.6.1].

Next, we verify $\eta(R) = \mathcal{R}$. First, suppose $(w, v) = (u_1 v_1 \cdots u_n v_n, v_1 \cdots v_n) \in R$ with the properties from above. From (iii), we obtain $u_1 v_1 \cdots u_n v_n \sim u_1 u_2 \cdots u_n\, v_1 v_2 \cdots v_n$ and therefore $([w], [v]) = ([u_1 \cdots u_n][v], [v])$ which belongs to $\mathcal{R}$ since $u_1 \cdots u_n \in K$ by (i). Thus, $\eta(R) \subseteq \mathcal{R}$. Conversely, let $([uv], [v]) \in \mathcal{R}$, i.e., $u \in K$ and $v \in \Sigma^*$. With $n = 1$, $u_1 = u$, and $v_1 = v$, we get $(uv, v) = (u_1 v_1, v_1) \in R$ and therefore $\mathcal{R} \subseteq \eta(R)$. Thus, indeed, $\eta(R) = \mathcal{R}$.

It remains to be shown that R is left-closed. So let $n \in \mathbb{N}$ and $u_1, \ldots, u_n, v_1, \ldots, v_n \in \Sigma^*$ satisfying (i-iii) from above and let $w \in \Sigma^*$ such that

$$w \sim u_1 v_1 u_2 v_2 \cdots u_n v_n \; R \; v_1 \cdots v_n\,.$$

With $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_n$, (iii) implies

$$w \sim u_1 v_1 u_2 v_2 \cdots u_n v_n \sim u_1 \cdots u_n \, v_1 \cdots v_n = uv \,.$$

Applying Levi's Lemma for traces [11, p. 74] to the equivalence $w \sim uv$ yields $m \in \mathbb{N}$ and words $u'_1, \ldots, u'_m$ and $v'_1, \ldots, v'_m$ such that

(i) $w = u'_1 v'_1 \, u'_2 v'_2 \cdots u'_m v'_m$,
(ii) $u \sim u'_1 u'_2 \cdots u'_m =: u'$,
(iii) $v \sim v'_1 v'_2 \cdots v'_m =: v'$, and
(iv) $(v'_1 v'_2 \cdots v'_i, u'_{i+1}) \in I$ for all $i \in [m-1]$.

Note that $u' \sim u \in K = \eta^{-1}(\mathcal{K})$ implies $u' \in K$. Hence we get $w \; R \; v' \sim v$. Thus, indeed, the relation R is left-closed. ◀

▶ **Lemma 4.11.** *Let $\mathcal{L} \subseteq \mathbb{M}$ be rational. Then $\mathcal{R} = (\{[\varepsilon]\} \times \mathcal{L}) \cdot \mathrm{Id}_{\mathbb{M}} = \{(y, xy) \mid x \in \mathcal{L}, y \in \mathbb{M}\}$ is lc-rational.*

Proof. There exists a regular language $L \subseteq \Sigma^*$ with $\eta(L) = \mathcal{L}$. Furthermore, $R := \{(v, uv) \mid u \in L, v \in \Sigma^*\}$ is the product of $L \times \{\varepsilon\}$ and Id_{Σ^*} and therefore rational.

We first show that R is even lc-rational. So let $u \in L$ and $v \sim v'$ be arbitrary words such that $v \sim v' \; R \; uv'$. Then we have $v \; R \; uv \sim uv'$. Hence, indeed, R is left-closed.

Next we show $\mathcal{R} = \eta(R)$. For the inclusion "$\supseteq$", let $(v, uv) \in R$, i.e., $u \in L$ and $v \in \Sigma^*$. From $\eta(L) = \mathcal{L}$, we obtain $[u] \in \mathcal{L}$ and therefore $(\eta(v), \eta(uv)) = (\eta(v), \eta(u)\,\eta(v)) \in \mathcal{R}$. Thus, $\mathcal{R} \supseteq \eta(R)$. Conversely, let $(y, xy) \in \mathcal{R}$, i.e., $x \in \mathcal{L}$ and $y \in \mathbb{M}$. From $x \in \mathcal{L} = \eta(L)$, we obtain a word $u \in L$ with $\eta(u) = x$. Further, there is a word $v \in \Sigma^*$ with $\eta(v) = y$. It follows that $(v, uv) \in R$ and therefore $(y, xy) = (\eta(v), \eta(uv)) \in \eta(R)$. ◀

Now the following sufficient condition for the lc-rationality of $\mathcal{R}_1 \cdot \mathcal{R}_2$ follows:

▶ **Theorem 4.12.** *Let $\mathcal{K} \subseteq \mathbb{M}$ be recognizable, $\mathcal{L} \subseteq \mathbb{M}$ rational, and $\mathcal{R} \subseteq \mathbb{M}^2$ lc-rational. Then $(\mathcal{K} \times \mathcal{L}) \cdot \mathcal{R}$ is lc-rational.*

Proof. The relations $\mathcal{R}_1 = (\mathcal{K} \times \{[\varepsilon]\}) \cdot \mathrm{Id}_{\mathbb{M}}$ and $\mathcal{R}_2 = (\{[\varepsilon] \times \mathcal{L}\}) \cdot \mathrm{Id}_{\mathbb{M}}$ are lc-rational by Lemmas 4.10 and 4.11, respectively. Hence, by Proposition 4.5(ii), the relation $\mathcal{R}_1 \circ \mathcal{R} \circ \mathcal{R}_2$ is lc-rational as well. But this composition equals $(\mathcal{K} \times \mathcal{L}) \cdot \mathcal{R}$. ◀

5 An application of lc-rational relations

Inspired by asynchronous automata [36] for recognizable trace languages, we introduced "cooperating multi-pushdown systems" or cPDS, i.e., distributed pushdown systems whose pushdown stores a trace as opposed to a word as in classical pushdown systems [23]. Hence, sets of configurations are trace languages. As main result, we proved that backwards reachability in these cPDS preserves the recognizability and forwards reachability preserves the rationality of sets of configurations; these two proofs use very different proof techniques. It is the aim of this section to show that the reachability relation of a cPDS is an lc-rational trace relation which, by Theorem 4.7, gives the two main results from [23] in a uniform manner.

Before we can define cPDS, we need the following notation. Since Σ is finite, there exists a finite set J and a mapping $\lambda \colon \Sigma \to 2^J \setminus \{\emptyset\}$ such that $(a, b) \in D$ iff $\lambda(a) \cap \lambda(b) \neq \emptyset$.

A *cPDS* is a tuple $\mathbb{P} = (\prod_{i \in J} Q_i, \Delta)$ for some finite sets Q_i of local states and a finite set $\Delta \subseteq \prod_{i \in J} Q_i \times \Sigma \times \Sigma^* \times \prod_{i \in J} Q_i$ of transition rules such that the following hold (where $\overline{p}$ denotes a tuple $(p_i)_{i \in J} Q_i$):

(i) If $(\overline{p}, a, w, \overline{q}) \in \Delta$, then $p_i = q_i$ for all $i \in J \setminus \lambda(a)$.
(ii) If $(\overline{p}, a, w, \overline{q}) \in \Delta$ and $\overline{r} \in \prod_{i \in J} Q_i$ with $p_i = r_i$ for all $i \in \lambda(a)$, then

$$\Big(\overline{r}, a, w, \big((q_i)_{i \in \lambda(a)}, (r_i)_{i \notin \lambda(a)}\big)\Big) \in \Delta\,.$$

(iii) If $(\overline{p}, a, ubv, \overline{q}) \in \Delta$, then $\lambda(a) \supseteq \lambda(b)$.

A *configuration* of $\mathbb{P}$ is a pair $(\overline{p}, x)$ with $\overline{p} \in \prod_{i \in J} Q_i$ a global state and $x \in \mathbb{M}$ a trace. For two configurations $c_1 = (\overline{p}, x)$ and $c_2 = (\overline{q}, z)$, there is a transition from c_1 to c_2 (written $c_1 \vdash c_2$) iff there exists a transition rule $(\overline{p}, a, w, \overline{q}) \in \Delta$ and a trace $y \in \mathbb{M}$ such that $x = [a] \cdot y$ and $z = [w] \cdot y$. Combining results from [23] and from the present paper, we can derive the following claim.

▶ **Lemma 5.1.** *Let $\mathbb{P} = (\prod_{i \in J} Q_i, \Delta)$ be a cPDS and $\overline{p}, \overline{q} \in \prod_{i \in J} Q_i$ global states. The trace relation $\mathcal{R}_{\overline{p},\overline{q}} = \{(x, y) \in \mathbb{M} \times \mathbb{M} \mid (\overline{p}, x) \vdash^* (\overline{q}, y)\}$ is lc-rational.*

Proof. A cPDS is *saturated* if $(\overline{p}, a, ubv, \overline{q}), (\overline{q}, b, \varepsilon, \overline{r}) \in \Delta$ with $(u, b) \in I$ implies $(\overline{p}, a, uv, \overline{r}) \in \Delta$. In [23, Section 5.3], we show how to transform a cPDS $\mathbb{P}$ into a saturated $\mathbb{P}'$ that has the same reachability relation $\vdash^*$. Hence, we can assume $\mathbb{P}$ to be saturated.

From [23, Lemma 5.4], one obtains that the relation $\mathcal{R}_{\overline{p},\overline{q}}$ is a finite union of compositions of relations of the form $(\mathcal{K} \times \mathcal{L}) \cdot \mathrm{Id}_{\mathbb{M}}$ with $\mathcal{K} \subseteq \mathbb{M}$ recognizable and $\mathcal{L} \subseteq \mathbb{M}$ rational. Now Theorem 4.12 and Proposition 4.5 imply that the reachability relation $\mathcal{R}_{\overline{p},\overline{q}}$ is lc-rational. ◀

Using the above lemma and Theorem 4.7, we obtain the following results from [23] in a uniform manner.

▶ **Theorem 5.2** (Köcher and Kuske [23]). *Let $\mathbb{P} = (\prod_{i \in J} Q_i, \Delta)$ be a cPDS, $\overline{p}, \overline{q} \in \prod_{i \in J} Q_i$, and $\mathcal{L} \subseteq \mathbb{M}$.*

(i) *If $\mathcal{L}$ is recognizable, then $\mathcal{L}_{\mathrm{b}} := \{x \in \mathbb{M} \mid \exists y \in \mathcal{L} \colon (\overline{p}, x) \vdash^* (\overline{q}, y)\}$ is recognizable, i.e., backwards reachability in cPDS preserves recognizability [23, Thm. 4.1].*
(ii) *If $\mathcal{L}$ is rational, then $\mathcal{L}_{\mathrm{f}} := \{y \in \mathbb{M} \mid \exists x \in \mathcal{L} \colon (\overline{p}, x) \vdash^* (\overline{q}, y)\}$ is rational, i.e., forwards reachability in cPDS preserves rationality [23, Thm. 5.1].*

Proof. The claims follow from the observations $\mathcal{L}_{\mathrm{b}} = {}^{\mathcal{R}_{\overline{p},\overline{q}}}\mathcal{L}$ and $\mathcal{L}_{\mathrm{f}} = \mathcal{L}^{\mathcal{R}_{\overline{p},\overline{q}}}$. ◀

6 The theory of lc-rational trace relations II

6.1 A Nivat-type characterisation of lc-rational relations

There are three very basic types of rational word relations $R \subseteq \Sigma^* \times \Sigma^*$: (graphs of) homomorphisms $\Sigma^* \to \Sigma^*$, their inverses, and restrictions to regular languages K, i.e., relations of the form $R_K = \{(u, u) \mid u \in K\}$. Nivat's theorem [30] (cf. [4, Thm. III.3.2]) shows that every rational relation R can be composed of these three basic types. More precisely, he showed that any rational relation R can be written as $f^{-1} \circ R_K \circ g$ for a regular language K and homomorphisms f and g. This result does not only hold for rational relations on words, but for arbitrary monoids, in particular for rational relations $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$ [4, Prop. III.3.4]. Our next result builds on this characterisation of all *rational relations* $\mathcal{R}$ and provides a similar characterisation of all *lc-rational relations* $\mathcal{R}$ in the spirit of Nivat's theorem.

▶ **Theorem 6.1.** *Let $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$. Then $\mathcal{R}$ is lc-rational if, and only if, there exists an alphabet Γ, a regular language $K \subseteq \Gamma^*$, and homomorphisms $\boldsymbol{f}, \boldsymbol{g} \colon \Gamma^* \to \mathbb{M}$ such that*
(a) *$\mathcal{R} = \{(\boldsymbol{f}(x), \boldsymbol{g}(x)) \mid x \in K\}$ and*

(b) *for all* $y \in K$, $n \in \mathbb{N}$, *and* $t_1, \ldots, t_n \in \mathbb{M}$ *with* $t_1 \cdot t_2 \cdots t_n = \boldsymbol{f}(y)$, *there exist* $y_1, y_2, \ldots, y_n \in \Gamma^*$ *such that*
- **(b1)** $y_1 y_2 \cdots y_n \in K$,
- **(b2)** $\boldsymbol{f}(y_i) = t_i$ *for all* $i \in [n]$, *and*
- **(b3)** $\boldsymbol{g}(y) = \boldsymbol{g}(y_1 \cdots y_n)$.

As explained above, the relation $\mathcal{R}$ is rational if, and only if, there are Γ, K, and $\boldsymbol{f}$, $\boldsymbol{g}$ satisfying property (a). The left-closedness therefore corresponds to property (b).

Proof. The detailed proof can be found in the appendix.

First, suppose $\mathcal{R}$ is lc-rational, i.e., the η-image of some lc-rational word relation R. Let Γ, K, f, and g be the alphabet, regular language, and homomorphisms that exist by Theorem 3.1(R4). Then $\boldsymbol{f} = \eta \circ f$ and $\boldsymbol{g} = \eta \circ g$ are homomorphisms from Γ^* to $\mathbb{M}$ with the claimed properties.

Conversely, let Γ, K, $\boldsymbol{f}$, and $\boldsymbol{g}$ be the alphabet, regular language, and homomorphisms from the theorem. Then there are homomorphisms $f, g \colon \Gamma^* \to \Sigma^*$ with $\boldsymbol{f} = \eta \circ f$ and $\boldsymbol{g} = \eta \circ g$. Together with K, they define a rational word relation R. One then shows that R is left-closed and satisfies $\mathcal{R} = \eta(R)$. ◀

6.2 Undecidability

So far, we proved that lc-rational trace relations share some of the nice properties of rational word relations. Unfortunately, it is undecidable (not even semi-decidable) whether a given rational relation is left-closed.

▶ **Proposition 6.2.** *Suppose D is transitive and I is not transitive such that there are $a, b, c \in \Sigma$ with $(a, b), (b, c) \in I$ and $(a, c) \in D$. It is not semi-decidable whether a given rational trace relation $\mathcal{R} \subseteq \mathbb{M}^2$ is lc-rational.*

Proof. Let P be an instance of Post's correspondence problem. From P, Muscholl and Petersen [29, proof of Thm. 2] construct (following a similar construction in [33]) a star-free language $L \subseteq \Sigma^*$ such that $\overline{L} = \eta^{-1}\eta(L)$ is star-free iff P has no solution (this construction uses that I is not transitive). Since D is transitive, $\overline{L}$ is either star-free or not regular [29, Thm. 1]. Hence, $\overline{L}$ is regular if, and only if, P has no solution.

Recall that any star-free language L is regular such that $\mathcal{L} := \eta(L)$ is rational. Hence

$$\begin{aligned}
\mathcal{L} \times \{[\varepsilon]\} \text{ is lc-rational} &\iff \mathcal{L} \text{ is recognizable} && \text{by Proposition 4.6} \\
&\iff \eta^{-1}(\mathcal{L}) \text{ is regular} \\
&\iff \overline{L} \text{ is regular} && \text{since } \eta^{-1}(\mathcal{L}) = \overline{L} \\
&\iff P \text{ has no solution}\,.
\end{aligned}$$

Thus, one can construct a rational relation $\mathcal{L} \times \{[\varepsilon]\}$ from P that is lc-rational if, and only if, P has no solution. Since the existence of *no* solution is not semi-decidable, the claim follows. ◀

7 Rational trace relations

We saw that the composition of a rational and an lc-rational trace relation is rational, again. This holds in particular if the first relation is the inverse of an lc-rational relation. We now demonstrate that all rational trace relations arise in this way.

▶ **Theorem 7.1.** *Suppose there are $a, b \in \Sigma$ with $(a, b) \in D$. Let $\mathcal{R} \subseteq \mathbb{M}^2$ be a binary relation. Then the following are equivalent:*

(i) *$\mathcal{R}$ is rational.*

(ii) *There exist lc-rational relations $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $\mathcal{R} = \mathcal{R}_1^{-1} \circ \mathcal{R}_2$.*

Proof. The implication "(ii)⇒(i)" is immediate by Proposition 4.5(i) since $\mathcal{R}_1^{-1}$ is rational for any rational relation $\mathcal{R}_1$.

For the converse implication, suppose $\mathcal{R}$ is rational. Then there exists a rational relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\mathcal{R} = \eta(R)$. By Nivat's theorem, i.e., Theorem 3.1(R4), there exist an alphabet Γ, homomorphisms f and g from Γ^* to Σ^*, and a regular language $K \subseteq \Gamma^*$ such that

$$R = \{(f(u), g(u)) \mid u \in K\}\,.$$

Suppose $\Gamma = \{c_1, c_2, \ldots, c_n\}$. Let $h\colon \Gamma^* \to \Sigma^*$ be the homomorphisms defined by $h(c_i) = a^i b$.

Now consider the relations

$$R_1 = \{(h(u), f(u)) \mid u \in K\} \text{ and } R_2 = \{(h(u), g(u)) \mid u \in K\}\,.$$

We first show that these relations are lc-rational (by symmetry, we only consider the relation R_1). From Nivat's theorem, we obtain that R_1 is rational. To show that it is left-closed, let $v, v', w' \in \Sigma^*$ with $v \sim v'\ R_1\ w'$.

Since $R_1 \subseteq \{a, b\}^* \times \Sigma^*$, we obtain $v' \in \{a, b\}^*$. Since $(a, b) \in D$, this implies $v = v'$. Hence, setting $w := w'$, we obtain $v\ R_1\ w \sim w'$. Hence, indeed, the relations R_1 and R_2 are lc-rational.

Next, we show $R = R_1^{-1} \circ R_2$. For the inclusion "$\subseteq$", let $(v, w) \in R$. Then there exists $u \in K$ with $v = f(u)$ and $w = g(u)$. Hence we obtain

$$v = f(u)\ R_1^{-1}\ h(u)\ R_2\ g(u) = w$$

and therefore $(v, w) \in R_1^{-1} \circ R_2$. For the converse inclusion, suppose $(v, w) \in R_1^{-1} \circ R_2$. Then there exists some word x with $v\ R_1^{-1}\ x\ R_2\ w$. By the definition of the relations R_1 and R_2, there are words $u_1, u_2 \in K \subseteq \Gamma^*$ such that

$$v = f(u_1)\,, x = h(u_1) \text{ and } x = h(u_2)\,, w = g(u_2)\,.$$

Since the homomorphism h is injective, we get $u_1 = u_2$ and therefore

$$(v, w) = (f(u_1), g(u_2)) = (f(u_1), g(u_1)) \in R\,.$$

Thus, indeed, $R = R_1^{-1} \circ R_2$.

Finally, let $\mathcal{R}_1 = \eta(R_1)$ and $\mathcal{R}_2 = \eta(R_2)$. Note that R_1^{-1} is rational and satisfies $\eta(R_1^{-1}) = \mathcal{R}_1^{-1}$. From Proposition 3.5(i), we obtain that

$$\eta(R_1^{-1} \circ R_2) = \eta(R_1^{-1}) \circ \eta(R_2) = \eta(R_1^{-1}) \circ \mathcal{R}_2$$

since R_2 is lc-rational. Hence, we obtain

$$\mathcal{R} = \eta(R) = \eta(R_1^{-1} \circ R_2) = \eta(R_1^{-1}) \circ \mathcal{R}_2 = \mathcal{R}_1^{-1} \circ \mathcal{R}_2\,. \qquad ◀$$

Recall that the right-application of rational trace relations does neither preserve the rationality nor the recognizability of a set of traces. The above factorisation result allows to prove that the right-application transforms recognizable sets into rational ones.

▶ **Theorem 7.2.** *Let $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$ be rational and $\mathcal{L} \subseteq \mathbb{M}$ recognizable. Then $\mathcal{L}^{\mathcal{R}}$ is rational.*

Proof. By Theorem 7.1, there are lc-rational relations $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $\mathcal{R} = \mathcal{R}_1^{-1} \circ \mathcal{R}_2$. Hence $\mathcal{L}^{\mathcal{R}} = \mathcal{L}^{\mathcal{R}_1^{-1} \circ \mathcal{R}_2} = (\mathcal{L}^{\mathcal{R}_1^{-1}})^{\mathcal{R}_2} = ({}^{\mathcal{R}_1}\mathcal{L})^{\mathcal{R}_2}$. From Theorem 4.7, we know that $\mathcal{K} := {}^{\mathcal{R}_1}\mathcal{L}$ is recognizable and therefore in particular rational. Hence, again using Theorem 4.7, $\mathcal{K}^{\mathcal{R}_2}$ is rational. ◀

Note that the above lemma cannot be improved: by Lemma 4.9, there exist an lc-rational relation $\mathcal{R}$ and a recognizable set $\mathcal{L}$ such that $\mathcal{L}^{\mathcal{R}}$ is not recognizable. Furthermore, there is also a rational set $\mathcal{L}$ such that ${}^{\mathcal{R}}\mathcal{L}$ is not rational. Note that ${}^{\mathcal{R}}\mathcal{L} = \mathcal{L}^{\mathcal{R}^{-1}}$ and that $\mathcal{R}^{-1}$ is rational. Hence, indeed, the above lemma is optimal.

▶ **Proposition 7.3.** *The relation $\leq_{\text{rat}}$ is not transitive.*

Proof. Consider the lc-rational relation $\mathcal{R}$ and the languages $\mathcal{K}$ and $\mathcal{L}$ from the proof of Lemma 4.9. In that proof, we showed in particular that $\mathcal{L}$ is recognizable, $\mathcal{L}^{\mathcal{R}} = \mathcal{K}$, and ${}^{\mathcal{R}}\mathcal{K}$ is not rational. Hence, by Theorem 7.2, there is no rational trace relation $\mathcal{S}$ with $\mathcal{L}^{\mathcal{S}} = {}^{\mathcal{R}}\mathcal{K}$, i.e., ${}^{\mathcal{R}}\mathcal{K} \not\leq_{\text{rat}} \mathcal{L}$. On the other hand, we have ${}^{\mathcal{R}}\mathcal{K} = \mathcal{K}^{\mathcal{R}^{-1}} \leq_{\text{rat}} \mathcal{K}$ since also $\mathcal{R}^{-1}$ is a rational relation as well as $\mathcal{K} = \mathcal{L}^{\mathcal{R}} \leq \mathcal{L}$. ◀

From this proposition, we get immediately that rational trace relations do not compose for otherwise $\leq_{\text{rat}}$ would be transitive (Example 3.2 provides concrete rational trace relations whose composition is not rational).

Thus, rational trace relations enjoy properties (R3) and (R4) from Theorem 3.1, but not the properties (R1) and (R2). On the other hand, lc-rational relations

- satisfy some versions of (R1) (cf. Theorem 4.7), but not all natural versions (cf. Lemma 4.9),
- satisfy (R2) (cf. Proposition 4.5),
- violate (R3), and
- have a characterisation in the spirit of (R4) (cf. Theorem 6.1).

We next restrict the class of lc-rational trace relations further to obtain a class that satisfies all versions of (R1), (R2), (R3), and has a characterisation in the spirit of (R4).

First, a word relation $R \subseteq \Sigma^* \times \Sigma^*$ is *right-closed* if $\sim \circ R \supseteq R \circ \sim$, i.e., for all $u, v' \in \Sigma^*$, we have

$$\left(\exists u' \in \Sigma^* \colon u \sim u' \; R \; v'\right) \impliedby \left(\exists v \in \Sigma^* \colon u \; R \; v \sim v'\right).$$

Compared with the definition of left-closedness, we only inverted the inclusion and implication, resp. The relation R is *rc-rational* if it is right-closed and rational. A trace relation $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$ is *rc-rational*, if there exists some rc-rational relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\mathcal{R} = \eta(R)$.

Let $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$. We call $\mathcal{R}$ *closed-rational* if it is both, lc-rational and rc-rational. To prove closed-rationality of a relation $\mathcal{R}$, one has to produce two relations R_1 and R_2 with $\mathcal{R} = \eta(R_1) = \eta(R_2)$ that are lc-rational and rc-rational, resp. In particular, these two relations can be distinct.

▶ **Example 7.4.** With the notions from Lemma 4.10, the relation $\mathcal{R} = \{(xy, y) \mid x \in \mathcal{L}, y \in \mathbb{M}\}$ is lc-rational. Since any recognizable trace language $\mathcal{L}$ is also rational, the relation $\mathcal{R}^{-1} = \{(y, xy) \mid x \in \mathcal{L}, y \in \mathbb{M}\}$ is lc-rational by Lemma 4.11 implying that $\mathcal{R}$ is rc-rational. Hence $\mathcal{R}$ is even closed-rational.

Looking at the proofs of Lemmas 4.11 and 4.10, we see that the relations $R \subseteq \Sigma^* \times \Sigma^*$ used there are indeed very different.

The following lemma shows that we can certify closed-rationality by providing a single word relation R that is both, lc- and rc-rational.

▶ **Lemma 7.5.** *A relation $\mathcal{R} \subseteq \mathbb{M}^2$ is closed-rational iff there exists a single relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\eta(R) = \mathcal{R}$ that is both, lc-rational and rc-rational.*

Proof. The implication "$\Leftarrow$" is immediate.

So let $\mathcal{R}$ be closed-rational. Then there are relations R_1 and R_2 with $\mathcal{R} = \eta(R_1) = \eta(R_2)$ that are lc-rational and rc-rational, resp. One then shows that $R = R_1 \cup R_2$ is left- and right-closed (see appendix) which implies the claim since $\eta(R) = \mathcal{R}$ and R is rational. ◀

Based on this characterisation, the proof of Theorem 6.1 can be extended easily showing the following Nivat-type characterisation of all closed-rational relations:

▶ **Theorem 7.6.** *Let $\mathcal{R} \subseteq \mathbb{M} \times \mathbb{M}$. Then $\mathcal{R}$ is closed-rational if, and only if, there exists an alphabet Γ, a regular language $K \subseteq \Gamma^*$, and homomorphisms $\boldsymbol{f}, \boldsymbol{g} \colon \Gamma^* \to \mathbb{M}$ such that*

(a) $\mathcal{R} = \{(\boldsymbol{f}(x), \boldsymbol{g}(x)) \mid x \in K\}$,

(b) *for all $y \in K$, $n \in \mathbb{N}$, and $t_1, \dots, t_n \in \mathbb{M}$ with $t_1 \cdot t_2 \cdots t_n = \boldsymbol{f}(y)$, there exist $y_1, y_2, \dots, y_n \in \Gamma^*$ such that*
  - **(b1)** $y_1 y_2 \cdots y_n \in K$,
  - **(b2)** $\boldsymbol{f}(y_i) = t_i$ *for all* $i \in [n]$, *and*
  - **(b3)** $\boldsymbol{g}(y) = \boldsymbol{g}(y_1 \cdots y_n)$, *and*

(c) *for all $y \in K$, $n \in \mathbb{N}$, and $t_1, \dots, t_n \in \mathbb{M}$ with $t_1 \cdot t_2 \cdots t_n = \boldsymbol{g}(y)$, there exist $y_1, y_2, \dots, y_n \in \Gamma^*$ such that*
  - **(c1)** $y_1 y_2 \cdots y_n \in K$,
  - **(c2)** $\boldsymbol{g}(y_i) = t_i$ *for all* $i \in [n]$, *and*
  - **(c3)** $\boldsymbol{f}(y) = \boldsymbol{f}(y_1 \cdots y_n)$.

Finally, the results of this paper show that the class of closed-rational relations has all of the properties from Theorem 3.1 that make rational relations on Σ^* interesting:

▶ **Theorem 7.7.** *The class of closed-rational relations on $\mathbb{M}$ is closed under inversion, under composition, and any closed-rational relation preserves both, rationality and recognizability under both, left- and right-application.*

8 Conclusion

We restricted the class of rational trace relation in order to get a notion that has properties similar to those of rational word relations. Our findings were applied in order to improve the central result on systems with a trace pushdown from [23].

For the definition of lc- and closed-rational relations, it was crucial that the trace monoid $\mathbb{M}$ was finitely generated by the alphabet Σ. Left-closedness can similarly be defined for any monoid M generated by some set Σ as follows. Let $\eta \colon \Sigma^* \to M$ be the natural epimorphism. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is left-closed if, for any $u, v' \in \Sigma^*$, we have

$$(\exists u' \in \Sigma^* \colon \eta(u) = \eta(u') \text{ and } (u', v') \in R) \implies (\exists v \in \Sigma^* \colon (u, v) \in R \text{ and } \eta(v) = \eta(v'))\,.$$

Further, a relation $\mathcal{R} \subseteq M \times M$ is lc-rational if there exists a left-closed and rational relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\mathcal{R} = \eta(R)$. It is, at least mathematically, interesting which of our results on lc-rational relations transfer to this more general setting (actually, many proofs should go through without much alteration, but, e.g., the proof of Lemma 4.10 makes explicite use of Levi's lemma).

References

1. V. Bárány, E. Grädel, and S. Rubin. Automata-based presentations of infinite structures. In *Finite and Algorithmic Model Theory*, pages 1–76. Cambridge University Press, 2011.
2. P. Barceló, D. Figueira, and L. Libkin. Graph logics with rational relations and the generalized intersection problem. In *LICS'12*, pages 115–124. IEEE Computer Society, 2012.
3. W. Bekker and V. Goranko. Symbolic model checking of tense logics on rational Kripke models. In *ILC'07*, Lecture Notes in Comp. Sciecne, pages 2–20. Springer, 2009.
4. J. Berstel. *Transductions and context-free languages.* Teubner Studienbücher, Stuttgart, 1979.
5. B. Bollig. *Formal Models of Communicating Systems – Languages, Automata, and Monadic Second-Order Logic.* Springer, 2006.
6. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR'97*, Lecture Notes in Mathematics, vol. 1243. Springer, 1997.
7. A. Carayol and A. Meyer. Context-sensitive languages, rational graphs and determinism. *Log. Methods Comput. Sci.*, 2(2), 2006.
8. A. Carayol and C. Morvan. On rational trees. In *CSL'06*, Lecture Notes in Computer Science vol. 4207, pages 225–239. Springer, 2006.
9. P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements.* Lecture Notes in Mathematics vol. 85. Springer, Berlin – Heidelberg – New York, 1969.
10. D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.
11. V. Diekert and G. Rozenberg. *The Book of Traces.* World Scientific Publ. Co., 1995.
12. M. Droste, P. Gastin, and D. Kuske. Asynchronous cellular automata for pomsets. *Theoretical Computer Science*, 247:1–38, 2000. (Fundamental study).
13. M. Droste and D. Kuske. Recognizable and logically definable languages of infinite computations in concurrent automata. *International Journal of Foundations of Computer Science*, 9:295–314, 1998.
14. S. Eilenberg. *Automata, Languages and Machines vol. A.* Academic Press, New York, 1974.
15. C.C. Elgot and G. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Develop.*, 9:47–65, 1965.
16. A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. *Electronic Notes in Theoretical Computer Science*, 9:27–37, 1997.
17. Ch. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *Theoretical Computer Science*, 108:45–82, 1993.
18. E. Grädel. Automatic structures: Twenty years later. In *LICS '20*, pages 21–34. ACM, 2020.
19. J. H. Johnson. Rational equivalence relations. *Theor. Comput. Sci.*, 47(3):39–60, 1986.
20. J. H. Johnson. Uniformizing rational relations for natural language applications using weighted determinization. In *CIAA'10*, Lecture Notes in Computer Science vol. 6482, pages 173–180. Springer, 2010.
21. S.C. Kleene. Representation of events in nerve nets and finite automata. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, Annals of Mathematics Studies vol. 34, pages 3–40. Princeton University Press, 1956.
22. Ch. Köcher. *Verification of Automata with Storage Mechanisms.* TU Ilmenau, 2022. Doctoral dissertation.
23. Ch. Köcher and D. Kuske. Forwards- and backwards-reachability for cooperating multi-pushdown systems. In *FCT'23*, Lecture Notes in Comp. Science vol. 14252, pages 318–332. Springer, 2023.
24. D. Kuske. The subtrace order and counting first-order logic. In *CSR'20*, Lecture Notes in Comp. Science vol. 12159, pages 289–302. Springer, 2020.
25. D. Kuske and A. Muscholl. Communicating automata. In J.-E. Pin, editor, *Handbook of Automata Theory*, volume 2, pages 1147–1188. EMS Press, 2021.

26 A. Mazurkiewicz. Concurrent program schemes and their interpretation. Technical report, DAIMI Report PB-78, Aarhus University, 1977.

27 C. Morvan. On rational graphs. In *FOSSACS'00*, Lecture Notes in Comp. Science vol. 1784, pages 252–266. Springer, 2000.

28 C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In *MFCS'01*, Lecture Notes in Comp. Science vol. 2136, pages 548–559. Springer, 2001.

29 A. Muscholl and H. Petersen. A note on the commutative closure of star-free languages. *Inf. Process. Lett.*, 57(2):71–74, 1996.

30 M. Nivat. Transductions des langages de Chomsky. *Ann. Inst. Fourier*, 18(1):339–456, 1968.

31 J.-E. Pin. Finite automata. In J.-E. Pin, editor, *Handbook of Automata Theory*, pages 3–38. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.

32 Sasha Rubin. Automatic structures. In J.-E. Pin, editor, *Handbook of Automata Theory*, pages 1031–1070. EMS Press, 2021.

33 J. Sakarovitch. The "last" decision problem for rational trace languages. In *LATIN'92*, Lecture Notes in Comp. Science vol. 583, pages 460–473. Springer, 1992.

34 J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

35 W. Thomas. A short introduction to infinite automata. In *DLT'01*, Lecture Notes in Comp. Science vol. 2295, pages 130–144. Springer, 2002.

36 W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. – Informatique Théorique et Applications*, 21:99–135, 1987.

A Appendix

Case distinction from Example 3.4.

- Suppose $x = u_1v_1 \cdots u_{i-1}v_{i-1}x'$, $u_i = x'bay'$, and $y = y'v_iu_{i+1}v_{i+1} \cdots u_{n+1}$. Then

$$\begin{aligned} u &= u_1v_1u_2 \cdots u_{i-1}v_{i-1}\, x'aby'\, v_iu_{i+1}v_{i+1} \cdot u_{n+1} \\ &\succeq v_1v_2 \cdots v_n = v' \,. \end{aligned}$$

  Thus, setting $v := v'$ yields $u \succeq v \sim v'$.
- Suppose $x = u_1v_1 \cdots u_{i-1}v_{i-1}u_ix'$, $v_i = x'bay'$, and $y = y'u_{i+1}v_{i+1} \cdots u_{n+1}$. Then

$$\begin{aligned} u &= u_1v_1 \cdots u_{i-1}v_{i-1}u_i\, x'aby'\, u_{i+1}v_{i+1} \cdots v_nu_{n+1} \\ &\succeq v_1 \cdots v_{i-1}\, x'aby'\, v_{i+1} \cdots v_n \\ &\sim v_1 \cdots v_{i-1}\, v_i\, v_{i+1} \cdots v_n = v' \,. \end{aligned}$$

  Thus, setting $v := v_1 \cdots v_{i-1}\, x'aby'\, v_{i+1} \cdots v_n$ yields $u \succeq v \sim v'$.
- Suppose $x = u_1v_1 \cdots u_{i-1}v_{i-1}x'$, $u_i = x'b$, $v_i = ay'$, and $y = y'u_{i+1}v_{i+1} \cdots u_{n+1}$. Then

$$\begin{aligned} u &= u_1v_1 \cdots u_{i-1}v_{i-1}\, x'b\, ay'\, u_{i+1}v_{i+1} \cdots u_{n+1} \\ &\succeq v_1 \cdots v_{i-1}\, ay'\, v_{i+1} \cdots v_n = v' \,. \end{aligned}$$

  Thus, setting $v := v'$ yields $u \succeq v \sim v'$.
- Finally, suppose $x = u_1v_1 \cdots u_ix'$, $v_i = x'b$, $u_{i+1} = ay'$, and $y = y'v_{i+1}u_{i+2}v_{i+2} \cdots u_{n+1}$. Then

$$\begin{aligned} u &= u_1v_1 \cdots u_i\, x'b\, ay'\, v_{i+1}u_{i+2}v_{i+2} \cdots u_{n+1} \\ &\succeq v_1 \cdots v_{i-1}x'bv_{i+1} \cdots v_{n+1} = v \,. \end{aligned}$$

  Thus, setting $v := v'$ yields $u \succeq v \sim v'$. ◀

Proof of first claim of Prop. 3.6. Let $K \times L$ be rational. Then the mapping $\pi_1 \colon \Sigma^* \times \Sigma^* \to \Sigma^* \colon (u, v) \mapsto u$ is a homomorphism and maps the rational set $K \times L$ onto the set K which is therefore rational, i.e., regular since it is a subset of Σ^*. The analogous argument leads to the regularity of L.

Conversely, suppose K and L both to be regular. Then $K \times \{\varepsilon\}$ is the image of the rational set $K \subseteq \Sigma^*$ under the homomorphism $f \colon \Sigma^* \to \Sigma^* \times \Sigma^* \colon u \mapsto (u, \varepsilon)$ and therefore a rational relation. The same applies to $\{\varepsilon\} \times L$. Since

$$K \times L = (K \times \{\varepsilon\}) \cdot (\{\varepsilon\} \times L)\,,$$

the relation $K \times L$ is the product of two rational relations and therefore rational. ◀

So first the counter-example.

▶ **Lemma A.1.** *There exist lc-rational relations $\mathcal{R}$ and $\mathcal{R}'$ such that $\mathcal{R} \cdot \mathcal{R}'$ is not lc-rational.*

Proof. Consider the rational trace relations $\mathcal{R}_1$ and $\mathcal{R}_2$ from Example 3.2. Note that $\mathcal{R}_2$ is the product of the lc-rational trace relations $\mathcal{R} = \{([d^n], [b^n]) \mid n \geq 1\}$ and $\mathcal{R}' = \{([c^m], [a^m]) \mid m \geq 1\}$ and recall that $\mathcal{R}_1 \circ \mathcal{R}_2$ is not rational. Hence, by Proposition 4.5(i), $\mathcal{R}_2$ cannot be lc-rational. ◀

Proof of $\mathcal{R}_1 \circ \mathcal{R} \circ \mathcal{R}_2 = (\mathcal{K} \times \mathcal{L}) \cdot \mathcal{R}$ in proof of Theorem 4.12. Note that $(x, z) \in \mathcal{R}_1 \circ \mathcal{R} \circ \mathcal{R}_2$ iff there are $y_1, y_2 \in \mathbb{M}$ with $(x, y_1) \in \mathcal{R}_1$, $(y_1, y_2) \in \mathcal{R}$, and $(y_2, z) \in \mathcal{R}_2$. But $(x, y_1) \in \mathcal{R}_1$ is equivalent to the existence of $k \in \mathcal{K}$ with $x = k \cdot y_1$. Similarly, $(y_2, z) \in \mathcal{R}_2$ iff there is $\ell \in \mathcal{L}$ with $z = \ell \cdot y_2$. In summary, we have $(x, z) \in \mathcal{R}_1 \circ \mathcal{R} \circ \mathcal{R}_2$ iff there exist $k \in \mathcal{K}$, $(y_1, y_2) \in \mathcal{R}$, and $\ell \in \mathcal{L}$ with $(x, z) = (k\, y_1, \ell\, y_2)$. But this holds iff $(x, z) \in (\mathcal{K} \times \mathcal{L}) \cdot \mathcal{R}$. ◀

Proof of Theorem 6.1. First, suppose $\mathcal{R}$ is lc-rational. Then there exists an lc-rational relation $R \subseteq \Sigma^* \times \Sigma^*$ with $\mathcal{R} = \eta(R)$. Hence, by Nivat's theorem [30] (cf. [4, Thm. III.3.2]), there exists an alphabet Γ, a regular language $K \subseteq \Gamma^*$, and homomorphisms $f, g \colon \Gamma^* \to \Sigma^*$ such that

$$R = \{(f(x), g(x)) \mid x \in K\}\,.$$

Because of the equivalence of (i) and (iii) in [4, Thm. III.3.2], we can even assume that the homomorphisms f and g are *alphabetic*, i.e., $|f(a)| \leq 1$ and $|g(a)| \leq 1$ for all $a \in \Gamma$. Let $\boldsymbol{f} = \eta \circ f \colon \Gamma^* \to \mathbb{M}$ and $\boldsymbol{g} = \eta \circ g$. From $\mathcal{R} = \eta(R)$, we immediately get property (a). To also show (b), let $y \in K$, $n \in \mathbb{N}$, and $t_1, \ldots, t_n \in \mathbb{M}$ with $t_1 t_2 \cdots t_n = \boldsymbol{f}(y)$. Then there are words $u_1, \ldots, u_n \in \Sigma^*$ with $\eta(u_i) = t_i$ for all $i \in [n]$. Note that

$$u_1 u_2 \cdots u_n \sim f(y)\; R\; g(y)$$

since $\eta(u_1 u_2 \cdots u_n) = \boldsymbol{f}(y) = \eta(f(y))$. Since the relation R is left-closed, there exists a word $v \in \Sigma^*$ such that

$$u_1 \cdots u_n\; R\; v \sim g(y)\,.$$

It follows that there is $y' \in K$ with

$$u_1 u_2 \cdots u_n = f(y')\; R\; g(y') = v \sim g(y)\,.$$

Since the homomorphism f is alphabetic, we can split the word y' into

$$y' = y_1 y_2 \cdots y_n \text{ with } u_i = f(y_i) \text{ for all } i \in [n]\,.$$

We can now verify conditions (b1–3):

(b1) $y_1 \cdots y_n = y' \in K$,
(b2) $\boldsymbol{f}(y_i) = \eta(f(y_i)) = \eta(u_i) = t_i$ for all $i \in [n]$, and
(b3) $\boldsymbol{g}(y) = \eta(g(y)) \overset{v \sim g(y)}{=} \eta(v) = \eta(g(y')) = \boldsymbol{g}(y') = \boldsymbol{g}(y_1 \cdots y_n)$
Thus, we have indeed an alphabet Γ, a regular language K, and homomorphisms $\boldsymbol{f}$ and $\boldsymbol{g}$ such that (a) and (b) hold.

Conversely, suppose Γ is an alphabet, $K \subseteq \Gamma^*$ a regular language, and $\boldsymbol{f}, \boldsymbol{g} \colon \Gamma^* \to \mathbb{M}$ homomorphisms such that (a) and (b) hold.

Choose, for every letter $x \in \Gamma$, words $v_x, w_x \in \Sigma^*$ with $\boldsymbol{f}(x) = \eta(v_x)$ and $\boldsymbol{g}(x) = \eta(w_x)$. Let $f, g \colon \Gamma^* \to \Sigma^*$ be the homomorphisms with $f(x) = v_x$ and $g(x) = w_x$ for all $x \in \Gamma$. Then we obtain $\boldsymbol{f} = \eta \circ f$ and $\boldsymbol{g} = \eta \circ g$. Let

$$R = \{(f(x), g(x)) \mid x \in L\} \subseteq \Sigma^* \times \Sigma^* .$$

From Nivat's theorem, we obtain that R is a rational relation. By its very definition and property (a), we also get $\mathcal{R} = \eta(R)$.

It remains to be shown that the word relation R is left-closed. So let $u, u', v' \in \Sigma^*$ with

$$u \sim u' \, R \, v' .$$

With $n = |u|$, there are letters $a_i \in \Sigma$ for all $i \in [n]$ with $u = a_1 a_2 \cdots a_n$. For $i \in [n]$, we set $t_i = \eta(a_i)$. From $(u', v') \in R$, we obtain some word $y \in K$ with $u' = f(y)$ and $v' = g(y)$. Note that

$$\boldsymbol{f}(y) = \eta(u') = \eta(a_1 \cdots a_n) = t_1 \cdot t_2 \cdots t_n .$$

From (b), we obtain words $y_1, \ldots, y_n \in \Gamma^*$ such that
(b1) $y_1 \cdots y_n \in K$,
(b2) $\boldsymbol{f}(y_i) = t_i$ for all $i \in [n]$, and
(b3) $\boldsymbol{g}(y) = \boldsymbol{g}(y_1 \cdots y_n)$.
Since a_i is the only word with $t_i = \eta(a_i)$, property (b2) implies $f(y_i) = a_i$ for all $i \in [n]$. We therefore get

$$\begin{aligned} u = a_1 \cdots a_n = f(y_1 \cdots y_n) \, R \, g(y_1 \cdots y_n) & \qquad \text{by property (b1)} \\ \sim g(y) & \qquad \text{by property (b3)} \\ = v' . \end{aligned}$$

With $v = g(y_1 \cdots y_n)$, we therefore get

$$u \, R \, v \sim v' .$$

Hence R is left-closed which completes the proof of the lc-rationality of $\mathcal{R}$. ◀

Proof of Lemma 7.5. The implication "$\Leftarrow$" is immediate.

So let $\mathcal{R}$ be closed-rational. Then there are relations R_1 and R_2 with $\mathcal{R} = \eta(R_1) = \eta(R_2)$ that are lc-rational and rc-rational, resp. Set $R = R_1 \cup R_2$. Clearly, this relation is rational. We show that it is left-closed (the proof of its right-closedness is symmetric). So let $u, u', v' \in \Sigma^*$ with

$$u \sim u' \, (R_1 \cup R_2) \, v' .$$

If $(u', v') \in R_1$, the left-closedness of R_1 yields some word v with $u \; R_1 \; v \sim v'$ and therefore $u \; R \; v \sim v'$. So suppose $(u', v') \in R_2$. Since $\eta(R_2) = \mathcal{R} = \eta(R_1)$, there exists $(u'', v'') \in R_1$ with $u' \sim u''$ and $v' \sim v''$. Hence we get

$$u \sim u' \sim u'' \; R_1 \; v'' \sim v' \, .$$

Now the left-closedness of R_1 yields a word v with $u \; R_1 \; v \sim v'' \sim v'$ and therefore $u \; R \; v \sim v'$. ◀

Towards a Practical, Budget-Oblivious Algorithm for the Adwords Problem Under Small Bids

Vijay V. Vazirani ✉
University of California, Irvine, CA, USA

Abstract

Motivated by recent insights into the online bipartite matching problem (OBM), our goal was to extend the optimal algorithm for it, namely Ranking, all the way to the special case of adwords problem, called Small, in which bids are small compared to budgets; the latter has been of considerable practical significance in ad auctions [20]. The attractive feature of our approach was that it would yield a *budget-oblivious algorithm*, i.e., the algorithm would not need to know budgets of advertisers and therefore could be used in autobidding platforms.

We were successful in obtaining an optimal, budget-oblivious algorithm for Single-Valued, under which each advertiser can make bids of one value only. However, our next extension, to Small, failed because of a fundamental reason, namely failure of the *No-Surpassing Property*. Since the probabilistic ideas underlying our algorithm are quite substantial, we have stated them formally, after assuming the No-Surpassing Property, and we leave the open problem of removing this assumption.

With the help of two undergrads, we conducted extensive experiments on our algorithm on randomly generated instances. Our findings are that the No-Surpassing Property fails less than 2% of the time and that the performance of our algorithms for Single-Valued and Small are comparable to that of [20]. If further experiments confirm this, our algorithm may be useful as such in practice, especially because of its budget-obliviousness.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Adwords problem, ad auctions, online bipartite matching, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.21

Related Version *Full Version*: `https://arxiv.org/pdf/2107.10777.pdf`

Funding *Vijay V. Vazirani*: Supported in part by NSF grant CCF-2230414.

1 Introduction

The *adwords problem*, called Adwords[1] in this paper, involves matching keyword queries, as they arrive online, to advertisers; the latter have daily budget limits and they make bids for the queries. Its special case when bids are small compared to budgets, called Small in this paper, captures a key computational issue that arises in the context of ad auctions, for instance in Google's AdWords marketplace. An optimal algorithm for Small, achieving a competitive ratio of $\left(1-\frac{1}{e}\right)$, was first given in [20]; for the impact of this result in the marketplace, see Section 1.2. *In this paper, we give a new budget-oblivious online algorithm for Small.*

A *budget-oblivious online algorithm* does not know the daily budgets of advertisers; however, in a run, it knows when the budget of an advertiser is exhausted. Yet its revenue is compared to the optimal revenue generated by an offline algorithm with full knowledge of the budget. The importance of a budget-oblivious algorithm lies in its use in autobidding

[1] For formal statements of problems studied in this paper, see Section 2

© Vijay V. Vazirani;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 21; pp. 21:1–21:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

platforms [1, 6], which manage the ad campaigns of large advertisers; they dynamically adjust bids and budgets over multiple search engines to improve performance. In Open Problem Number 20, Mehta [19] asks for such an algorithm for Small.

Recent insights on the online bipartite matching problem (OBM) encouraged us to seek such an algorithm. A simple optimal algorithm, called Ranking, achieving a competitive ratio of $\left(1 - \frac{1}{e}\right)$, was given in [17] for OBM. However, the analysis of Ranking given in [17] was difficult to comprehend. A sequence of papers has finally led to a simple and elegant analysis, see Section 1.1. The simplicity of Ranking is particularly attractive; moreover, it has become the paradigm-setting algorithmic idea in the area of online and matching-based market design [8].

Ideas underlying the new proof of OBM enabled us to generalize Ranking all the way to an algorithm for Small, while retaining the simplicity of the Ranking. As a result of this simplicity, our algorithm has better properties than [20]; in particular, it is budget-oblivious. A detailed discussion of its running time is given below. A budget-oblivious algorithm for Small, having a competitive ratio of 0.522[2] was recently obtained by Udwani [21], using the idea of an LP-free analysis, which involves writing appropriate linear inequalities to compare the online algorithm with the offline optimal algorithm.

At the outset of this work, extending Ranking directly to Small seemed an uphill task. Therefore we attempted an intermediate problem first, namely Single-Valued, in which each advertiser can make bids of one value only, although the value may be different for different advertisers. We note that [2] had already obtained an optimal online algorithm for Single-Valued by reducing it to the vertex weighted online matching problem, see Section 1.1 for details. As explained in Section 1.3, in order to develop tools for attacking Small, we needed to solve Single-Valued *directly*, and not resort to this reduction.

Our algorithm for Single-Valued is optimal, and it is also budget-oblivious. Furthermore, our algorithm uses fewer random bits than the approach of [2]; see Section 1.1 for a detailed comparison. We note that in contemporary[3] and independent work, Albers and Schubert [3] obtained an identical result for Single-Valued; their technique is different and involves formulating a configuration LP and conducting a primal-dual analysis. Our technical ideas are described in Section 1.3.

Our analysis of Single-Valued involved new ideas from two domains, namely probability theory and combinatorics, with the former playing a dominant role and the latter yielding a proof of a condition called the *No-Surpassing Property*, see Property 11. Equipped with these ideas, we next attempted an extension from Ranking to Small. Although we met with success in extending the more difficult, probabilistic part, of the argument, we found a counter-example to the combinatorial part, showing that the No-Surpassing Property does not hold for Small.

In order to make the no-surpassing property fail, we had to intricately "doctored up" the instance of Small. This raised the question of experimentally determining how often this property fails in typical instances and how it affects the performance of our algorithm; for the latter, we compared it to [20]. As can be seen in the four Tables in the full paper [24], the property fails rarely, for less than 2% of the edges (i, j), and the performance of our algorithms for Single-Valued and Small are comparable with that of the MSVV Algorithm. For this reason, and because of its budget-obliviousness, the algorithm may be useful as such in practice. Clearly, it will be good to obtain further experimental confirm on varied types of instances.

[2] Note that the greedy algorithm, which is clearly budget-oblivious, achieves a competitive ratio of 0.5.
[3] Our paper was first posted on arXiv on July 22, 2021 [22].

Since the ideas underlying our algorithm for SMALL, and the probabilistic part of its proof, are quite substantial, we have stated them formally, after assuming the No-Surpassing Property, see the full paper [24]. Under this assumption, we prove a competitive ratio of $\left(1-\frac{1}{e}\right)$ for our algorithm. The problem of obtaining a tight unconditional competitive ratio of our algorithm is an important one and has received much attention over the last two years, ever since the appearance of this paper on arXiv. Critical insights into this open problem are provided by the following results: first, Udwani [21] gave an example to show that the unconditional competitive ratio of our algorithm is strictly less than $(1-1/e)$. Next, Liang et al. [18] showed that the unconditional competitive ratio is less than 0.624; in contrast, $(1-1/e) \approx 0.632$.

▶ Remark 1. The objective of all problems studied in this paper is to maximize the total revenue accrued by the online algorithm. In economics, such a solution is referred to as *efficient*, since the amount bid by an advertiser is indicative of how useful the query is to it, and hence to the economy.

1.1 Related Works

OBM occupies a central place not only in online algorithms but also in matching-based market design, see details in Section 1.2. The analysis of RANKING given in [17] was considered "difficult" and it also had an error. Over the years, several researchers contributed valuable ideas to simplifying its proof. The first simplifications, in [11, 4], got the ball rolling, setting the stage for the substantial simplification given in [7], using a randomized primal-dual approach. [7] introduced the idea of splitting the contribution of each matched edge into primal and dual contributions and lower-bounding each part separately. Their method for defining prices p_j of goods, using randomization, was used by subsequent papers, including this one[4].

Interestingly enough, the next simplification involved removing the scaffolding of LP-duality and casting the proof in purely probabilistic terms[5], using notions from economics to split the contribution of each matched edge into the contributions of the buyer and the seller. This elegant analysis was given by [9]. We note that when we move to generalizations of OBM, even this economic interpretation needs to be dropped, see Remark 4. Building on these works, and incorporating a further simplification relating to the No-Surpassing Property for OBM, a "textbook quality" proof was recently given in [23].

An important generalization of OBM is online b-matching. This problem is a special case of ADWORDS in which the budget of each advertiser is $\$b$ and the bids are 0/1. [16] gave a simple optimal online algorithm, called BALANCE, for this problem. BALANCE awards the next query to the interested bidder who has been matched least number of times so far. [16] showed that as b tends to infinity, the competitive ratio of BALANCE tends to $\left(1-\frac{1}{e}\right)$.

Observe that b-matching is a special case of SMALL, if b is large. Indeed, MSVV Algorithm was obtained by extending BALANCE[6] as follows: [20] first gave a simpler proof of the competitive ratio of BALANCE using the notion of a *factor-revealing LP* [15]. Then they

[4] For a succinct proof of optimality of the underlying function, e^{x-1}, see Section 2.1.1 in [12].
[5] Even though there is no overt use of LP-duality in the proof of [9], it is unclear if this proof could have been obtained directly, without going the LP-duality-route.
[6] It is worth recalling that [20] had first attempted extending OBM to SMALL; however, in the absence of new insights into OBM, this did not go very far.

gave the notion of a *tradeoff-revealing LP*, which yielded an algorithm achieving a competitive ratio of $\left(1-\frac{1}{e}\right)$. [20] also proved that this is optimal for b-matching, and hence Small, by proving that no randomized algorithm can achieve a better ratio for online b-matching; previously, [16] had shown a similar result for deterministic algorithms.

The MSVV Algorithm is simple and operates as follows. The effective bid of each bidder j for a query is its bid multiplied by $(1-e^{L_j/B_j})$, where B_j and L_j are the total budget and the leftover budget of bidder j, respectively; the query is matched to the bidder whose effective bid is highest. As a result, the MSVV Algorithm needs to know the total budget of each bidder. Following [20], a second optimal online algorithm for Small was given in [5], using a primal-dual approach.

Another relevant generalization of OBM is online vertex weighted matching, in which the offline vertices have weights and the objective is to maximize the weight of the matched vertices. [2] extended Ranking to obtain an optimal online algorithm for this problem. Clearly, Single-Valued is intermediate between Adwords and online vertex weighted matching. [2] gave an optimal online algorithm for Single-Valued by reducing it to online vertex weighted matching. This involved creating k_j copies of each advertiser j. As a result, their algorithm needs to use $\sum_{j\in A} k_j$ random numbers, where A is the set of advertisers. On the other hand, our algorithm, and that of [3], needs to use only $|A|$ numbers.

Adwords is a notoriously difficult problem, partly due to its inherent structural difficulties, which are described in the full paper [24]. For Adwords, the greedy algorithm, which matches each query to the highest bidder, achieves a competitive ratio of 1/2. Until recently, that was the best possible. In [13] a marginally improved algorithm, with a ratio of 0.5016, was given. It is important to point out that this 60-page paper was a tour-de-force, drawing on a diverse collection of ideas – a testament to the difficulty of this problem.

In the decade following the conference version (FOCS 2005) of [20], search engine companies generously invested in research on models derived from OBM and adwords. The reason was two-fold: the substantial impact of [20] and the emergence of a rich collection of digital ad tools. It will be impossible to do justice to this substantial body of work, involving both algorithmic and game-theoretic ideas; for a start, see the surveys [19, 12].

1.2 Significance and Practical Impact

Google's AdWords marketplace generates multi-billion dollar revenues annually and the current annual worldwide spending on digital advertising is almost half a trillion dollars. These revenues of Google and other Internet services companies enable them to offer crucial services, such as search, email, videos, news, apps, maps etc. for free – services that have virtually transformed our lives.

We note that Small is the most relevant case of adwords for the search ads marketplace e.g., see [6]. A remarkable feature of Google, and other search engines, is the speed with which they are able to show search results, often in milliseconds. In order to show ads at the same speed, together with search results, the solution for Small needed to be minimalistic in its use of computing power, memory and communication.

The MSVV Algorithm satisfied these criteria and therefore had substantial impact in this marketplace. Furthermore, the idea underlying their algorithm was extracted into a simple heuristic, called *bid scaling*, which uses even less computation and is widely used by search engine companies today. As mentioned above, our Conditional Algorithm for Small is even more elementary and is budget-oblivious.

It will be useful to view the AdWords marketplace in the context of a bigger revolution, namely the advent of the Internet and mobile computing, and the consequent resurgence of the area of matching-based market design. The birth of this area goes back to the seminal

1962 paper of Gale and Shapley on stable matching [10]. Over the decades, this area became known for its highly successful applications, having economic as well as sociological impact. These included matching medical interns to hospitals, students to schools in large cities, and kidney exchange.

The resurgence led to a host of highly innovative and impactful applications. Besides the AdWords marketplace, which matches queries to advertisers, these include Uber, matching drivers to riders; Upwork, matching employers to workers; and Tinder, matching people to each other, see [14, 8] for more details.

A successful launch of such markets calls for economic and game-theoretic insights, together with algorithmic ideas. The Gale-Shapley Deferred Acceptance Algorithm and its follow-up works provided the algorithmic backbone for the "first life" of matching-based market design. The algorithm Ranking has become the paradigm-setting algorithmic idea in the "second life" of this area [8]. Interestingly enough, this result was obtained in the pre-Internet days, over thirty years ago.

1.3 Technical Ideas

Our extension from Ranking to Small needs to go via Adwords. It turns out that Adwords suffers from an inherent structural difficulty, see the full paper [24]. We temporarily finesse this difficulty by using the idea of "fake" money. The expected revenue of our online algorithm for Adwords is at least $(1 - 1/e)$ fraction of the optimal offline revenue; however, this total revenue consists of real as well as fake money. We provide an upper-bound on the fake money in the worst case, and this suffices to show that, asymptotically, the fake money used by Small, is negligible. Determining the true competitive ratio of our algorithm for Adwords is left as an interesting and important open problem.

As described in Section 1.1, Single-Valued can be reduced to online vertex weighted matching, by making k_j copies of each advertiser j; however, this reduction does not work for Adwords. The reason is that the manner in which budget B_j of bidder j gets partitioned into bids is not predictable in the latter problem; it depends on the queries, their order of arrival and the randomization executed in a run of the algorithm. Therefore, in order to build techniques to attack Adwords, we will first need to solve Single-Valued *without* reducing it to online vertex weighted matching.

This is done in Algorithm 1. Almost all of our new ideas, on the probabilistic front, needed to attack Small were obtained in the process analyzing this algorithm. First, since vertex j is not split into k_j copies, we cannot talk about the contribution of edges anymore. Even worse, we don't have individual vertices for keeping track of the revenue accrued from each match, as per the scheme of [9]. Our algorithm gets around this difficulty by accumulating revenue in the same "account" each time bidder j gets matched. The corresponding random variable, r_j, is called the *total revenue* of bidder j, for want of a better name, see Remark 4. Lower bounding $\mathbb{E}[r_j]$ is much more tricky than lower bounding the revenue of a good in OBM, since it involves "teasing apart" the k_j accumulations made into this account; this is done in Lemma 14.

The key fact needed in the analysis of Ranking is that for each edge $e = (i, j)$ in the underlying graph, its expected contribution to the matching produced is at least $(1 - 1/e)$. For this purpose, the random variable, u_e, called *threshold*, is defined in [23].

For analyzing Single-Valued, a replacement is needed for this lemma. For this purpose, we give the notion of a *j-star*, denoted X_j, which consists of bidder j together with edges to k_j of its neighbors in G, see Definition 10. The contribution of j-star X_j, is denoted by $\mathbb{E}[X_j]$, which is also defined in Definition 10. Finally, using the lower bound on $\mathbb{E}[r_j]$, Lemma 14 gives a lower $\mathbb{E}[X_j]$ for *every* j-star, X_j. This lemma crucially uses a new random variable, called *truncated threshold*, see Definition 9.

Next, we explain the reason for truncation in the definition of this random variable. Consider bidder j and a query i_l that is desired by j. Observe that in run $\mathcal{R}_j$[7], query i_l can get a bid as large as $B \cdot (1 - \frac{1}{e})$, where $B = \max_{k \in A}\{b_k\}$, whereas the largest bid that j can make to i_l is $b_j \cdot (1 - \frac{1}{e})$; in general, b_j may be smaller than B. Now, i_l contributes revenue to r_j only if i_l is matched to j in run $\mathcal{R}$, an event which will definitely not happen if $u_{e_l} > b_j \cdot (1 - \frac{1}{e})$. Therefore, whenever $u_{e_l} \in [b_j \cdot (1 - \frac{1}{e}),\ B \cdot (1 - \frac{1}{e})]$, the contribution to r_j is zero. By truncating u_{e_l} to $b_j \cdot (1 - \frac{1}{e})$, we have effectively changed the probability density function of u_{e_l} so that the probability of the event $u_{e_l} \in [b_j \cdot (1 - \frac{1}{e}), B \cdot (1 - \frac{1}{e})]$ is now concentrated at the event $u_{e_l} = b_j \cdot (1 - \frac{1}{e})$. From the viewpoint of lower bounding the revenue accrued in r_j, the two probability density functions are equivalent since the revenue accrued is zero under both these events. On the other hand, the truncated random variable enables us to apply the law of total expectation, in the proof of Lemma 14, in the same way as it was done in [23], without introducing more difficulties.

Finally, in order to establish the no-surpassing property for Single-Valued, we give the necessary combinatorial facts in Lemma 7 and Corollary 8. These facts are enhanced versions of the facts needed to prove the no-surpassing property for Ranking in [23].

2 Preliminaries

Online Bipartite Matching (OBM). Let B be a set of n buyers and S a set of n goods. A bipartite graph $G = (B, S, E)$ is specified on vertex sets B and S, and edge set E, where for $i \in B,\ j \in S$, $(i, j) \in E$ if and only if buyer i *likes* good j. G is assumed to have a perfect matching and therefore each buyer can be given a unique good she likes. Graph G is revealed in the following manner. The n goods are known up-front. On the other hand, the buyers arrive one at a time, and when buyer i arrives, the edges incident at i are revealed.

We are required to design an online algorithm $\mathcal{A}$ in the following sense. At the moment a buyer i arrives, the algorithm needs to match i to one of its unmatched neighbors, if any; if all of i's neighbors are matched, i remains unmatched. The difficulty is that the algorithm does not "know" the edges incident at buyers which will arrive in the future and yet the size of the matching produced by the algorithm will be compared to the best *off-line matching*; the latter of course is a perfect matching. The formal measure for the algorithm is defined in Section 2.1.

Adwords Problem (ADWORDS). Let A be a set of m *advertisers*, also called *bidders*, and Q be a set of n *queries*. A bipartite graph $G = (Q, A, E)$ is specified on vertex sets Q and A, and edge set E, where for $i \in Q$ and $j \in A$, $(i, j) \in E$ if and only if bidder j is *interested in* query i. Each query i needs to be matched[8] to at most one bidder who is interested in it. For each edge (i, j), bidder j *knows* his bid for i, denoted by $\text{bid}(i, j) \in \mathbb{Z}_+$. Each bidder also has a *budget* $B_j \in \mathbb{Z}_+$ which satisfies $B_j \geq \text{bid}(i, j)$, for each edge (i, j) incident at j.

Graph G is revealed in the following manner. The m bidders are known up-front and the queries arrive one at a time. When query i arrives, the edges incident at i are revealed, together with the bids associated with these edges. If i gets matched to j, then the matched edge (i, j) is assigned a weight of $\text{bid}(i, j)$. The constraint on j is that the total weight of matched edges incident at it be at most B_j. The objective is to maximize the total weight of all matched edges at all bidders.

[7] Run $\mathcal{R}_j$ is defined in Definition 6.
[8] Clearly, this is not a matching in the usual sense, since a bidder may be matched to several queries.

Adwords under Single-Valued Bidders (SINGLE-VALUED). SINGLE-VALUED is a special case of ADWORDS in which each bidder j will make bids of a single value, $b_j \in \mathbb{Z}_+$, for the queries he is interested in. If i accepts j's bid, then i will be matched to j and the weight of this matched edge will be b_j. Corresponding to each bidder j, we are also given $k_j \in \mathbb{Z}_+$, the maximum number of times j can be matched to queries. The objective is to maximize the total weight of matched edges. Observe that the matching M found in G is a b-matching with the b-value of each query i being 1 and of advertiser j being k_j.

Adwords under Small Bids (SMALL). SMALL is a special case of ADWORDS in which for each bidder j, each bid of j is small compared to its budget. Formally, we will capture this condition by imposing the following constraint. For a valid instance I of SMALL, define

$$\mu(I) = \max_{j \in A} \left\{ \frac{\max_{(i,j)\in E} \{\text{bid}(i,j) - 1\}}{B_j} \right\}.$$

Then we require that

$$\lim_{n(I)\to\infty} \mu(I) = 0,$$

where $n(I)$ denotes the number of queries in instance I.

2.1 The competitive ratio of online algorithms

We will define the notion of competitive ratio of a randomized online algorithm in the context of OBM.

▶ **Definition 2.** *Let $G = (B, S, E)$ be a bipartite graph as specified above. The competitive ratio of a randomized algorithm $\mathcal{A}$ for OBM is defined to be:*

$$c(\mathcal{A}) = \min_{G=(B,S,E)} \min_{\rho(B)} \frac{\mathbb{E}[\mathcal{A}(G, \rho(B))]}{n},$$

where $\mathbb{E}[\mathcal{A}(G, \rho(B))]$ is the expected size of matching produced by $\mathcal{A}$; the expectation is over the random bits used by $\mathcal{A}$. We may assume that the worst case graph and the order of arrival of buyers, given by $\rho(B)$, are chosen by an adversary who knows the algorithm. It is important to note that the algorithm is provided random bits after *the adversary makes its choices.*

▶ Remark 3. For each problem studied in this paper, we will assume that the offline matching is complete. It is easy to extend the arguments, without changing the competitive ratio, in case the offline matching is not complete.

3 Algorithm for Single-Valued

Algorithm 1, which will be denoted by $\mathcal{A}_1$, is an online algorithm for SINGLE-VALUED. Before execution of Step (1) of $\mathcal{A}_1$, the order of arrival of queries, say $\rho(B)$, is fixed by the adversary. We will define several random variables whose purpose will be quite similar to that in RANKING and they will be given similar names as well; however, their function is not as closely tied to these economics-motivated names as in RANKING, see also Remark 4. Three of these random variables are the *price* p_j and *total revenue* r_j of each bidder $j \in A$, and the *utility* u_i of each query $i \in Q$.

We now describe how values are assigned to these random variables in a run of Algorithm 1. In Step (1), for each bidder j, $\mathcal{A}_1$ picks a price $p_j \in [\frac{1}{e}, 1]$ via the specified randomized process. Furthermore, the revenue r_j and *degree* d_j of bidder j are both initialized to zero, the latter represents the number of times j has been matched. During the run of $\mathcal{A}_1$, j will get matched to at most k_j queries; each match will add b_j to the total revenue generated by the algorithm. b_j is broken into a revenue and a utility component, with the former being added to r_j and the latter forming u_i. At the end of $\mathcal{A}_1$, r_j will contain all the revenue accrued by j.

In Step (2), on the arrival of query i, we will say that bidder j is *available* if $(i, j) \in E$ and $d_j < k_j$. At this point, for each available bidder j, the *effective bid* of j for i is defined to be $\text{ebid}(j) = b_j \cdot (1 - p_j)$; clearly, $\text{ebid}(j) \in [0, b_j \cdot \left(1 - \frac{1}{e}\right)]$. Query i accepts the bidder whose effective bid is the largest. If there are no bids, matching M remains unchanged. If i accepts j's bid, then edge (i, j) is added to matching M and the weight of this edge is set to b_j. Furthermore, the *utility* of i, u_i, is defined to be $\text{ebid}(j)$ and the revenue r_j of j is incremented by $b_j \cdot p_j$. Once all queries are processed, matching M and its weight W are output.

▶ Remark 4. [9] had given the economics-based names of random variables for their proof of Ranking. Although we have used the same names for similar random variables in Section 3 for Single-Valued, the reader should not attribute an economic interpretation to these the names[9].

3.1 Analysis of Algorithm 1

For the analysis of Algorithm $\mathcal{A}_1$, we will use the random variables W, p_j, r_j and u_i defined above; their values are fixed during the execution of $\mathcal{A}_2$. In addition, corresponding to each edge $e = (i, j) \in E$, in Definition 9, we will introduce a new random variable, u_e, which will play a central role.

▶ **Lemma 5.**

$$\mathbb{E}[W] \;= \sum_{i}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j}^{m} \mathbb{E}[r_j].$$

Proof. For each edge $(i, j) \in M$, its contribution to W is b_j. Furthermore, the sum of u_i and the contribution of (i, j) to r_j is also b_j. This gives the first equality below. The second equality follows from linearity of expectation.

$$\mathbb{E}[W] \;= \mathbb{E}\left[\sum_{i=1}^{n} u_i \;+\; \sum_{j=1}^{m} r_j\right] \;= \sum_{i}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j}^{m} \mathbb{E}[r_j],$$ ◀

▶ **Definition 6.** *We will define several runs of Algorithm 1. In these runs, we will assume Step (1) is executed once. We next define several ways of executing Step (2). Let $\mathcal{R}$ denote the run of Step (2) on the entire graph G. Corresponding to each bidder $j \in A$, let G_j denote graph G with bidder j removed. Define $\mathcal{R}_j$ to be the run of Step (2) on graph G_j.*

[9] We failed to come up with more meaningful names for these random variables and therefore have stuck to the old names.

Algorithm 1 $\mathcal{A}_1$: Algorithm for SINGLE-VALUED.

1. **Initialization:** $M \leftarrow \emptyset$.
   $\forall j \in A$, do:
   a. Pick w_j uniformly from $[0, 1]$ and set price $p_j \leftarrow e^{w_j - 1}$.
   b. $r_j \leftarrow 0$.
   c. $d_j \leftarrow 0$.

2. **Query arrival:** When query i arrives, **do**:
   a. $\forall j \in A$ s.t. $(i, j) \in E$ and $d_j < k_j$ **do**:
      i. $\text{ebid}(j) \leftarrow b_j \cdot (1 - p_j)$.
      ii. Offer effective bid of $\text{ebid}(j)$ to i.
   b. Query i accepts the bidder whose effective bid is the largest.
      (If there are no bids, matching M remains unchanged.)
      If i accepts j's bid, then **do**:
      i. Set utility: $u_i \leftarrow b_j \cdot (1 - p_j)$.
      ii. Update revenue: $r_j \leftarrow r_j + b_j \cdot p_j$.
      iii. Update degree: $d_j \leftarrow d_j + 1$.
      iv. Update matching: $M \leftarrow M \cup (i, j)$. Define the weight of (i, j) to be b_j.
   c. **Output:** Output matching M and its total weight W.

Lemma 7 and Corollary 8 given below establish a relationship between the available bidders for a query i in the two runs $\mathcal{R}$ and $\mathcal{R}_j$. Note that bidders are available in multiplicity and therefore we will have to use the notion of a multiset rather than a set, as was done in [23].

A *multiset* contains elements with multiplicity. Let A and B be two multisets over n elements $\{1, 2, \ldots n\}$, and let $a_i \geq 0$ and $b_i \geq 0$ denote the multiplicities of element i in A and B, respectively. We will say that $A \subseteq B$ if for each i, $a_i \leq b_i$, and $A = B$ if for each i, $a_i = b_i$. We will say that $i \in A$ if $a_i \geq 1$. We will define $A \cap B$ to be the multiset containing each element i exactly $\min\{a_i, b_i\}$ times, and $A - B$ to be the multiset containing each element i exactly $\max\{a_i - b_i, 0\}$ times.

As before, let us renumber the queries so their order of arrival under $\rho(B)$ is $1, 2, \ldots n$. Let $T(i)$ and $T_j(i)$ denote the multisets of available bidders at the time of arrival of query i (i.e., just before the query i gets matched) in runs $\mathcal{R}$ and $\mathcal{R}_j$, respectively. In particular, $T(1)$ will contain k_l copies of l for each bidder l and $T_j(1)$ will contain k_l copies of l for each bidder l, other than j. Similarly, let $S(i)$ and $S_j(i)$ denote the projections of $T(i)$ and $T_j(i)$ on the neighbors of i in G and G_j, respectively.

We have assumed that Step (1) of Algorithm 1 has already been executed and a price p_k has been assigned to each bidder k. The effective bid of bidder k is $\text{ebid}(k) = b_k \cdot (1 - p_k)$. With probability 1, the effective bids of all bidders are distinct. Let F_1 be the multiset containing k_l copies of l for each $l \in A$ such that $b_l \cdot (1 - p_l) > b_j \cdot (1 - p_j)$. Similarly, let F_2 be the multiset containing k_l copies of l for each $l \in A$ such that and $b_l \cdot (1 - p_l) < b_j \cdot (1 - p_j)$. Observe that j is not contained in either multiset.

▶ **Lemma 7.** *For each i, $1 \leq i \leq n$, the following hold:*
1. $(T_j(i) \cap F_1) = (T(i) \cap F_1)$.
2. $(T_j(i) \cap F_2) \subseteq (T(i) \cap F_2)$.

Proof.

1. Clearly, in both runs, $\mathcal{R}$ and $\mathcal{R}_j$, any query having an available bidder in F_1 will match to the most profitable one of these, without even considering the rest of the bidders. Since $j \notin F_1$, the two runs behave in an identical manner on the set F_1, thereby proving the first statement.
2. The proof is by induction on i. The base case is trivially true because $(T_j(1) \cap F_2) = (T(1) \cap F_2)$, since $j \notin F_2$. Assume that the statement is true for $i = k$ and let us prove it for $i = k + 1$. By the first statement, we need to consider only the case that there are no available bidders for the kth query in F_1 in the runs $\mathcal{R}$ and $\mathcal{R}_j$. Assume that in run $\mathcal{R}_j$, this query gets matched to bidder l; if it remains unmatched, we will take l to be null. Clearly, l is the most profitable bidder it is incident to in $T_j(k)$. Therefore, the most profitable bidder it is incident to in run $\mathcal{R}$ is the best of l, the most profitable bidder in $T(k) - T_j(k)$, and j, in case it is available. In each of these cases, the induction step holds. ◀

In the corollary below, the first two statements follow from Lemma 7 and the third statement follows from the first two statements.

▶ **Corollary 8.** *For each i, $1 \leq i \leq n$, the following hold:*

1. $(S_j(i) \cap F_1) = (S(i) \cap F_1)$.
2. $(S_j(i) \cap F_2) \subseteq (S(i) \cap F_2)$.
3. $S_j(i) \subseteq S(i)$.

Next we define a new random variable, u_e, for each edge $e = (i, j) \in E$. This is called the *truncated threshold* for edge e and is given in Definition 9. It is critically used in the proofs of Lemmas 13 and 14.

▶ **Definition 9.** *Let $e = (i, j) \in E$ be an arbitrary edge in G. Define random variable, u_e, called the* truncated threshold *for edge e, to be $u_e = \min\{ut_i,\ b_j \cdot \left(1 - \frac{1}{e}\right)\}$, where ut_i is the utility of query i in run $\mathcal{R}_j$.*

▶ **Definition 10.** *Let $j \in A$. Henceforth, we will denote k_j by k in order to avoid triple subscripts. Let $i_1, \ldots, i_k$ be queries such that for $1 \leq l \leq k$, $(i_l, j) \in E$. Then $(j;\ i_1, \ldots, i_k)$ is called a j-*star*. Let X_j denote this j-star. The contribution of X_j to $\mathbb{E}[W]$ is $\mathbb{E}[r_j] + \sum_{l=1}^{k} \mathbb{E}[u_{i_l}]$, and it will be denote by $\mathbb{E}[X_j]$.*

Corresponding to j-star $X_j = (j;\ i_1, \ldots, i_k)$, denote by e_l the edge $(i_l, j) \in E$, for $1 \leq l \leq k$. Furthermore, let u_{e_l} denote the truncated threshold random variable corresponding to e_l.

▶ **Property 11** (No-Surpassing for Single-Valued)**.** *Assume that Step 1 of Algorithm 1 has been executed and a price p_k has been assigned to each advertiser k. Suppose that the effective bid which query i gets in run $\mathcal{R}_j$ is less than $b_j \cdot (1 - p_j)$; the latter is clearly the effective bid which j makes to i in run $\mathcal{R}$. Then, in run $\mathcal{R}$, no bid to i will surpass* $\text{ebid}(j) = b_j \cdot (1 - p_j)$.

▶ **Lemma 12.** *The No-Surpassing Property holds for Single-Valued.*

Proof. Suppose the bid of j, namely $b_j \cdot (1 - p_j)$, is better than the best bid that buyer i gets in run $\mathcal{R}_j$. If so, i gets no bid from F_1 in $\mathcal{R}_j$; observe that they are all higher than $b_j \cdot (1 - p_j)$. Now, by the first part of Corollary 8, i gets no bid from F_1 in run $\mathcal{R}$ as well, i.e., in run $\mathcal{R}$, no bid to i will surpass $b_j \cdot (1 - p_j)$. ◀

▶ **Lemma 13.** *Corresponding to j-star $X_j = (j;\ i_1, \ldots, i_k)$, the following hold.*
- *For $1 \le l \le k$, $u_{i_l} \ge u_{e_l}$.*

Proof. By the third statement of Corollary 8, i_l has more options in run $\mathcal{R}$ as compared to run $\mathcal{R}_j$. Furthermore, the truncation of the random variable only aids the inequality needed and therefore $u_{i_l} \ge u_{e_l}$. ◀

Our next goal is to lower bound the contribution of an arbitrary j-star, $\mathbb{E}[X_j]$, which in turn involves lower bounding $\mathbb{E}[r_j]$. The latter crucially uses the fact that p_j is independent of u_{e_l}. This follows from the fact that u_{e_l} is determined by run $\mathcal{R}_j$ on graph G_j, which does not contain vertex j.

▶ **Lemma 14.** *Let $j \in A$ and let $X_j = (j;\ i_1, \ldots, i_k)$ be a j-star. Then*

$$\mathbb{E}[X_j] \ \ge\ k \cdot b_j \cdot \left(1 - \frac{1}{e}\right).$$

Proof. We will first lower bound $\mathbb{E}[r_j]$. Let $f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)$ be the joint probability density function of $(u_{e_1}, \ldots u_{e_k})$; clearly, $f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)$ can be non-zero only if $z_l \in [0, 1 - \frac{1}{e}]$, for $1 \le l \le k$. By the law of total expectation,

$$\mathbb{E}[r_j] \ = \int_{(z_1,\ldots,z_k)} \mathbb{E}[r_j \mid u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k] \cdot f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)\ dz_1 \ldots dz_k,$$

where the integral is over $z_l \in [0, \left(1 - \frac{1}{e}\right)]$, for $1 \le l \le k$.

For lower-bounding the conditional expectation in this integral, let $w_l \in [0, 1]$ be s.t. $e^{w_l - 1} = 1 - z_l$, for $1 \le l \le k$. For $x \in [0, 1]$, define the set $S(x) = \{l \mid 1 \le l \le k \text{ and } x < w_l\}$.

▷ Claim 15. Conditioned on $(u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k)$, if $p_j = e^{x-1}$, then the degree of j at the end of Algorithm $\mathcal{A}_2$ is at least $|S(x)|$, i.e., the contribution to r_j in this run was $\ge b_j \cdot p_j \cdot |S(x)|$.

Proof. Suppose $l \in S(x)$, then $x < w_l$. In run $\mathcal{R}_j$, the maximum effective bid that i_l received has value $b_j \cdot z_l$. In run $\mathcal{R}$, if at the arrival of query i_l, j is already fully matched, the contribution to r_j in this run was $k \cdot b_j \cdot p_j$ and the claim is obviously true. If not, then since $x < w_l$, $b_j \cdot (1 - p_j) > b_j \cdot z_l$. The crux of the matter is that by Lemma 12, the No-Surpassing Property holds. Therefore, query i_l will receive its largest effective bid from j, i_l will get matched to it, and r_j will be incremented by $b_j \cdot p_j$. The claim follows. ◁

For $1 \le l \le k$, define indicator functions $I_l : [0, 1] \to \{0, 1\}$ as follows.

$$I_l(x) \ = \begin{cases} 1 & \text{if } x < w_l, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $|S(x)| = \sum_{l=1}^{k} I_j(x)$. By Claim 15,

$$\mathbb{E}[r_j \mid u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k] \ \ge\ b_j \cdot \int_0^1 |S(x)| \cdot e^{x-1}\ dx$$

$$= \ b_j \cdot \int_0^1 \sum_{l=1}^{k} I_l(x) \cdot e^{x-1}\ dx \ = \ b_j \cdot \sum_{l=1}^{k} \int_0^1 I_l(x) \cdot e^{x-1}\ dx \ = \ b_j \cdot \sum_{l=1}^{k} \int_0^{w_l} e^{x-1}\ dx$$

$$= \ b_j \cdot \sum_{l=1}^{k} \left(e^{w_l - 1} - \frac{1}{e}\right) \ = \ b_j \cdot \sum_{l=1}^{k} \left(1 - \frac{1}{e} - z_l\right).$$

Since $I_l(x) = 0$ for $x \in [w_l, 1]$, we get that $\int_0^1 I_l(x) \cdot e^{x-1}\, dx \;=\; \int_0^{w_l} e^{x-1}\, dx$; this fact has been used above. Therefore,

$$\mathbb{E}[r_j] \;=\int_{(z_1,\ldots,z_k)} \mathbb{E}[r_j \mid u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k] \cdot f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)\; dz_1 \ldots dz_k$$

$$\geq b_j \cdot \int_{(z_1,\ldots,z_k)} \sum_{l=1}^{k} \left(1 - \frac{1}{e} - z_l\right) \cdot f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)\; dz_1 \ldots dz_k$$

$$= \; k \cdot b_j \cdot \left(1 - \frac{1}{e}\right) - \sum_{l=1}^{k} \mathbb{E}[u_{e_l}],$$

where both integrals are over $z_l \in [0, \left(1 - \frac{1}{e}\right)]$, for $1 \leq l \leq k$.

By Lemma 13, $\mathbb{E}[u_{i_l}] \geq \mathbb{E}[u_{e_l}]$, for $1 \leq l \leq k$. Hence we get

$$\mathbb{E}[X_j] \;=\; \mathbb{E}[r_j] + \sum_{l=1}^{k} \mathbb{E}[u_{i_l}] \;\geq\; k \cdot b_j \cdot \left(1 - \frac{1}{e}\right),$$ ◀

▶ **Lemma 16.**

$$\mathbb{E}[W] \;= \sum_{i}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j}^{m} \mathbb{E}[r_j].$$

Proof. By definition of the random variables,

$$\mathbb{E}[W] \;= \mathbb{E}\left[\sum_{i=1}^{n} u_i \;+\; \sum_{j=1}^{m} r_j\right] \;= \sum_{i}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j}^{m} \mathbb{E}[r_j],$$

where the first equality follows from the fact that if $(i, j) \in M$ then W is incremented by b_j and $u_i + r_j = b_j$. The second equality follows from linearity of expectation. ◀

▶ **Theorem 17.** *The competitive ratio of Algorithm $\mathcal{A}_2$ is at least $1 - \frac{1}{e}$. Furthermore, it is budget-oblivious.*

Proof. Let P denote a maximum weight b-matching in G, computed in an offline manner. By the assumption made in Remark 3, its weight is

$$w(P) = \sum_{j=1}^{m} k_j \cdot b_j.$$

Let T_j denote the j-star, under P, corresponding to each $j \in A$. The expected weight of matching produced by $\mathcal{A}_2$ is

$$\mathbb{E}\left[W\right] \;=\; \sum_{i=1}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j=1}^{m} \mathbb{E}[r_j] \;=\; \sum_{j=1}^{m} \mathbb{E}[T_j] \;\geq\; \sum_{j=1}^{m} b_j \cdot k_j \left(1 - \frac{1}{e}\right) \;=\; \left(1 - \frac{1}{e}\right) \cdot w(P),$$

where the first equality uses Lemma 16, the second follows from linearity of expectation and the inequality follows from Lemma 14.

Finally, Algorithm $\mathcal{A}_2$ is budget-oblivious because it does not need to know k_j for bidders j; it only needs to know during a run whether the k_j bids available to bidder j have been exhausted. The theorem follows. ◀

References

1. Gagan Aggarwal, Ashwinkumar Badanidiyuru, and Aranyak Mehta. Autobidding with constraints. In *International Conference on Web and Internet Economics*, pages 17–30. Springer, 2019.
2. Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1253–1264, 2011.
3. Susanne Albers and Sebastian Schubert. Optimal algorithms for online b-matching with variable vertex capacities. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
4. Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM Sigact News*, 39(1):80–87, 2008.
5. Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264, 2007.
6. Nikhil Devanur and Aranyak Mehta. Online matching in advertisement auctions. In Federico Echenique, Nicole Immorlica, and Vijay V. Vazirani, editors, *Online and Matching-Based Market Design*. Cambridge University Press, 2022. [To appear] `https://www.ics.uci.edu/~vazirani/AdAuctions.pdf`.
7. Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 101–107. SIAM, 2013.
8. Federico Echenique, Nicole Immorlica, and Vijay V. Vazirani, editors. *Online and Matching-Based Market Design*. Cambridge University Press, 2023.
9. Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An economic-based analysis of ranking for online bipartite matching. In *SIAM Symposium on Simplicity in Algorithms*, 2021.
10. David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.
11. Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, volume 8, pages 982–991, 2008.
12. Zhiyi Huang and Thorben Trobst. Online matching. In Federico Echenique, Nicole Immorlica, and Vijay V. Vazirani, editors, *Online and Matching-Based Market Design*. Cambridge University Press, 2022. [To appear] `https://www.ics.uci.edu/~vazirani/Ch4.pdf`.
13. Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. Adwords in a panorama. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1416–1426. IEEE, 2020.
14. Simons Institute. Online and matching-based market design, 2019. URL: `https://simons.berkeley.edu/programs/market2019`.
15. Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM (JACM)*, 50(6):795–824, 2003.
16. Bala Kalyanasundaram and Kirk R Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319–325, 2000.
17. Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.
18. Jingxun Liang, Zhihao Gavin Tang, Yixuan Xu, Yuhao Zhang, and Renfei Zhou. On the perturbation function of ranking and balance for weighted online bipartite matching. *arXiv preprint*, 2022. `arXiv:2210.10370`.
19. Aranyak Mehta. *Online matching and ad allocation*, volume 8(4). Now Publishers, Inc., 2013.
20. Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5), 2007.

21 Rajan Udwani. Adwords with unknown budgets and beyond. *arXiv preprint*, 2021. `arXiv:2110.00504`.

22 Vijay V Vazirani. Online bipartite matching and adwords. *arXiv preprint*, 2021. `arXiv:2107.10777`.

23 Vijay V Vazirani. Online bipartite matching and adwords. In *47th International Symposium on Mathematical Foundations of Computer Science*, 2022.

24 Vijay V Vazirani. Towards a practical, budget-oblivious algorithm for the adwords problem under small bids. *arXiv preprint*, 2023. `arXiv:2107.10777`.

Languages Given by Finite Automata over the Unary Alphabet

Wojciech Czerwiński
Institute of Informatics, Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw, Poland

Maciej Dębski
Warsaw, Poland

Tomasz Gogasz
Institute of Informatics, Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw, Poland

Gordon Hoi
School of Informatics and IT, Temasek Polytechnic, Singapore, Singapore

Sanjay Jain
School of Computing, National University of Singapore, Singapore

Michał Skrzypczak
Institute of Informatics, Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw, Poland

Frank Stephan
Department of Mathematics and School of Computing,
National University of Singapore, Singapore

Christopher Tan
Department of Mathematics, National University of Singapore, Singapore

Abstract

This paper studies the complexity of operations on finite automata and the complexity of their decision problems when the alphabet is unary and n the number of states of the finite automata considered. The following main results are obtained:

(1) Equality and inclusion of NFAs can be decided within time $2^{O((n \log n)^{1/3})}$; previous upper bound $2^{O((n \log n)^{1/2})}$ was by Chrobak (1986) via DFA conversion.

(2) The state complexity of operations of UFAs (unambiguous finite automata) increases for complementation and union at most by quasipolynomial; however, for concatenation of two n-state UFAs, the worst case is an UFA of at least $2^{\Omega(n^{1/6})}$ states. Previously the upper bounds for complementation and union were exponential-type and this lower bound for concatenation is new.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Nondeterministic Finite Automata, Unambiguous Finite Automata, Upper Bounds on Runtime, Conditional Lower Bounds, Languages over the Unary Alphabet

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.22

Related Version *Full Version*: `https://arxiv.org/abs/2302.06435`

Funding *Wojciech Czerwiński*: supported by the ERC grant INFSYS, agreement no. 950398.
Sanjay Jain: supported by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2019-T2-2-121 / R146-000-304-112 as well as NUS Provost Chair grant E-252-00-0021-01.
Michał Skrzypczak: supported by the National Science Centre, Poland (grant number 2021/41/B/ST6/03914).
Frank Stephan: supported by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2019-T2-2-121 / R146-000-304-112.

© Wojciech Czerwiński, Maciej Dębski, Tomasz Gogasz, Gordon Hoi, Sanjay Jain, Michał Skrzypczak, Frank Stephan, and Christopher Tan;
licensed under Creative Commons License CC-BY 4.0

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 22; pp. 22:1–22:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

This paper investigates the complexity and size-constraints related to languages over the unary alphabet – this is assumed everywhere throughout the paper – when these languages are given by a nondeterministic finite automaton (NFA) with a special emphasis on the case where this NFA is unambiguous. Unambiguous nondeterministic finite automata (UFAs) have many good algorithmic properties, even under regular operations with the languages, as long as no concatenation is involved. The study of unary languages is a theoretically important special case which allows for techniques and insights from number theory, as each word corresponds to the natural number which is its length. This sometimes gives techniques which can be extended to general case. Furthermore, the case is a bit special as the NFA-DFA trade-off is only $2^{O((n \log n)^{1/2})}$ and not 2^n as for other alphabets; Colcombet [3] published several influential conjectures which found much attention: Göös, Kiefer and Yuan [6] refuted his conjecture that any two n-state NFAs with disjoint languages can be separated by an UFA of size polynomially in n; this paper confirms a weak version of another conjecture by showing that unary UFAs have only a quasipolynomial blow-up for Boolean operations; Raskin [18] had refuted the original version of the conjecture.

In the following, a bound of type $2^{\Theta(n)}$ is called exponential and a bound of type $2^{n^{\Theta(1)}}$ is called exponential-type. The quasipolynomial functions are those in $O(n^{\log^{O(1)}(n)})$ which are not bounded by polynomials. In an expression of the form $2^{\alpha(n)}$, the function $\alpha(n)$ is called the exponent of the function. Let $f \in \Omega'(g)$ mean that there is a constant $c > 0$ such that, for infinitely many n, $f(n) \geq c \cdot g(n)$. Note that, under the Exponential Time Hypothesis (ETH), all **NP**-complete problems have, for infinitely many inputs, an exponential-type complexity and solving kSAT with $k \geq 3$ requires time $2^{\Omega'(n)}$. For this paper, for several results, the exponential of an exponential-type function from the upper bound is matched, assuming Exponential Time Hypothesis, by a lower bound whose logarithm differs from that of the upper bound only by a logarithmic or sublogarithmic expression.

Assume two languages L, H are given by n-state NFAs and one wants to know whether $L \subseteq H$. Chrobak [2] provided an algorithm by converting the NFAs into DFAs and then comparing these which runs, for the unary alphabet, in time $2^{O((n \log n)^{1/2})}$. Fernau and Krebs [5] proved, under the assumption of ETH, the conditional lower bound $2^{\Omega'(n^{1/3})}$ but did not resolve the remaining gap. Tan provides in an undergraduate thesis (UROP) [21] an alternative proof for the conditional lower bound by coding the variant of SAT where all clauses have at most length 3 and all variables occur at most 3 times; this variant is called 3-occur-3SAT. The present work narrows the gap significantly by providing a comparison algorithm which decides whether $L \subseteq H$ in time $2^{O((n \log n)^{1/3})}$, see Theorem 2 below.

Recall that for an unambiguous nondeterministic finite automata (UFA), every word outside the language has zero accepting runs and every word inside the language has exactly one accepting run – see below for more explanations for these technical terms. Prior research had established that the intersection of two n-state UFAs can be represented by an $O(n^2)$-state UFA and that, over the binary alphabet, the complexity of the Kleene star of an n-state UFA can be recognised (synonymously accepted) by an $O(n^2)$-state UFA [1, 3, 8, 14, 16, 17]. But the size-increase of the other regular operations (complement, union, concatenation) remained open; it was however known that disjoint union has linear complexity.

Unambiguous finite automata found much attention in recent research with a lower quasipolynomial lower bound for the blow-up of the size for complementation by Raskin [18] of the form $\Omega'(n^{(\log \log \log n)^q})$ where q is some positive rational constant. Thus it is impossible to achieve a polynomial sized complementation of UFAs for any alphabet. For comparison with

the binary case, it should be mentioned that Göös, Kiefer and Yuan [6] proved that for binary languages of UFAs, the size of an NFA computing the complement is in $n^{\Omega'((\log n)/\mathrm{polylog}(\log n))}$. This lower bound also trivially applies to UFAs. However, this paper shows that, for unary alphabet, one can do all regular operations on UFAs with at most quasipolynomial size-increase except for concatenation where this paper achieves the lower bound of $2^{\Omega(n^{1/6})}$, confirming a weak version of a conjecture of Colcombet [3] – who predicted originally a polynomial size-increase for Boolean operations. For larger alphabets, there is a big gap between the lower bound $n^{\log n/\mathrm{polylog}(\log n)}$ by Göös, Kiefer and Yuan [6] and the upper bounds for complementing an n-state UFA of $2^{0.79n+\log n}$ by Jirásek, Jr., Jirásková and Šebej [13] and $\sqrt{n+1}\cdot 2^{n/2}$ by Indzhev and Kiefer [12].

For unary automata, Okhotin [15] determined the worst-case complexity on the determinisation of UFAs with $2^{\Theta((n\log^2(n))^{1/3})}$ and this is directly also an upper bound on the size-increase by complementation. In his master's thesis, author Dębski [4] constructed an algorithm to complement unambiguous automata over the unary alphabet while maintaining the upper size-bound of $n^{O(\log n)}$ for these automata. The present work tightens the analysis of the construction and gets the bound $n^{\log n+O(1)}$.

Note that it is not efficient to compare UFAs with respect to inclusion of the generated languages by constructing complementation of the second automaton and then taking intersection and checking for emptiness. Instead, one employs direct comparison algorithms; Stearns and Hunt [20] provided a polynomial time algorithm for this; this paper slightly improves it by providing a **LOGSPACE** algorithm in the case that the UFAs are in Chrobak Normal Form, and an **NLOGSPACE** algorithm without any assumption on a normal form. Note that the transformation into this normal form is a polynomial time algorithm which does not increase the number of states for UFAs; however, an **NLOGSPACE** algorithm might cause a polynomial size increase.

Tables 1, 2 and 3 provide the best known results by either this paper or prior work. Results of this paper refer to the corresponding theorem, proposition or remark while results of prior work give the corresponding reference. In particular, the tables summarise the results for UFAs. Previously published bounds are mentioned, where needed, near to the corresponding result. Note that $c(n) = n^{\log n+O(1)}$. Lower bounds on size imply lower bounds on computations; for concatenation a better lower bound is found (assuming ETH). In Table 1 the bounds on the size of UFAs are given. Table 2 (Table 3 respectively) give the bounds on the computation time (computation space respectively), needed for finding automata with the desired property or determining the truth of a formula are given. Note that in general for computing the complement, union, intersection and for comparing NFAs, the computation space $O(f(n))$ corresponds to the computation time $2^{O(f(n)}$ in the worst case. Thus every further improvement in the space usage will also have an improvement in the time usage. UFA comparison is in **NLOGSPACE**, which is more or less optimal and this corresponds to the polynomial time used for the comparison (though more concrete time bounds are given in the corresponding theorems).

2 Details of Technical Concepts and Methods

Finite Automata

A finite state automaton, see for example [9], is a tuple $(Q, \Sigma, Q_0, \delta, F)$, where Q is a finite set of states, Σ is a finite alphabet (unary for this paper), $Q_0 \subseteq Q$ is a set of initial states, δ is a transition function assigning to each pair in $Q \times \Sigma$ a subset of Q and $F \subseteq Q$ is a

Table 1 State complexity of operations with UFAs. Here $c(n) = n^{\log n + O(1)}$.

Operation	Lower Bound	Source	Upper Bound	Source
Intersection	$n^2 - n$	Holzer and Kutrib [7]	n^2	Holzer and Kutrib [7]
Complement	$n^{(\log\log\log n)^{\Omega(1)}}$	Raskin [18]	$c(n)$	Theorem 3
Disjoint union	$2n - 4$	–	$2n$	Jirásková and Okhotin [14]
Union	–	–	$n + n \cdot c(n)$	Remark 11
Symm. difference	$n^{(\log\log\log n)^{\Omega(1)}}$	Raskin [18]	$2n \cdot c(n)$	Theorem 3
Kleene Star	$(n-1)^2 + 1$	Čevorová [1]	$(n-1)^2 + 1$	Čevorová [1]
Concatenation	$2^{\Omega(n^{1/6})}$	Theorem 9	$2^{O((n \log^2 n)^{1/3})}$	Okhotin [15]

Table 2 Time bounds, except for the last, all refer to UFAs. The lower bounds using Ω' for computing concatenation, evaluating formulas using UFAs as inputs and comparing NFAs assume ETH and hold only infinitely often. The upper bound for universality in the first row assumes that the UFA is given in Chrobak Normal Form.

Operation	Lower Bound	Source	Upper Bound	Source
UFA Universality	–	–	$O(n \log^2 n)$	Theorem 5
UFA Comparison	–	–	Poly(n)	Stearns and Hunt III [20]
UFA Concatenation	$2^{\Omega'(n^{1/4})}$	Theorem 13	$2^{O((n \log^2 n)^{1/3})}$	Okhotin [15]
UFA Complement	$n^{(\log\log\log n)^{\Omega(1)}}$	Raskin [18]	$n^{O(\log n)}$	Theorem 3
UFA Formulas	$2^{\Omega'((n \log n)^{1/3})}$	Theorem 12	$2^{O((n \log^2 n)^{1/3})}$	Okhotin [15]
NFA Comparison	$2^{\Omega'(n^{1/3})}$	Fernau and Krebs [5]	$2^{O((n \log n)^{1/3})}$	Theorem 2

set of accepting states. A run of the automaton on input $a_1 a_2 \ldots a_n$ is a sequence of states $q_0, q_1, \ldots q_n$, such that $q_0 \in Q_0$ and for each $i < n$, $q_{i+1} \in \delta(q_i, a_{i+1})$. The run is accepting if $q_n \in F$. The input $a_1 a_2 \ldots a_n$ is accepted by the automaton if there is a run on it which is accepting. The set of words accepted by the automaton is called the language of the automaton. The terminologies word and string are used interchangeably.

Without further constraints, the automaton is called a *nondeterministic finite automaton* (NFA). An NFA accepts a word iff it has an accepting run for this word, that is, a word is accepted iff there is at least one run ending in an accepting state and a word is rejected iff there is either no run for the word at all or all runs for the word end in a rejecting state.

An NFA which has, for each word in its language L, exactly one accepting run and for each word outside L no accepting run is called an *unambiguous finite automaton* (UFA). An NFA which has exactly one start state and for which $\delta(q, a)$ has cardinality exactly one, for all $q \in Q$ and $a \in \Sigma$, is called a *deterministic finite automaton* (DFA). Note that every DFA has for each word exactly one run and this run is accepting iff the word is in the given language L. A language L is called *regular* iff L is the language of some finite automaton. The concept does not change if one requires this automaton to be a UFA or to be a DFA.

In the case of a unary alphabet one can bring an NFA or UFA into Chrobak Normal Form [2]. For the unary alphabet, a Chrobak Normal Form of the NFA is defined as follows: The NFA consists of states $q_1, q_2, \ldots, q_s$, $p_0^i, p_2^i, \ldots, p_{n_i-1}^i$, for i with $1 \leq i \leq r$, for some s, r and n_i, such that the following holds:

Table 3 Space Bounds; Result for Universality assumes input in Chrobak Normal Form and lower bound for NFA comparison assumes ETH and holds only infinitely often.

Operation	Lower Bound	Source	Upper Bound	Source
UFA Universality	–	–	$O(\log n)$	Theorem 5
NFA Comparison	ETH $\Rightarrow \Omega'(n^{1/3})$	Fernau and Krebs [5]	$O((n \log n)^{1/3})$	Theorem 2

(a) If $s > 0$ then q_1 is the starting state else $\{p_0^i : 1 \leq i \leq r\}$ is the set of starting states.
(b) $\delta(q_i, a) = \{q_{i+1}\}$, for i with $1 \leq i < s$ and $a \in \Sigma$.
(c) If $s > 0$ and $a \in \Sigma$, $\delta(q_s, a) = \{p_0^i : 1 \leq i \leq r\}$.
(d) $\delta(p_j^i, a) = \{p_{j+1 \mod n_i}^i\}$, for $a \in \Sigma$, $i \leq r$ and $j < n_i$.

The states $q_1, q_2, \ldots, q_s$ and the transitions within them is called the *stem* of the NFA, with q_s the last state of the stem. For $1 \leq i \leq r$, the states $(p_j^i)_{j<n_i}$ and the transitions among them, are called the *cycles* of the NFA, where n_i is the length of the cycle and p_0^i is the *entry state* of the cycle. Which states of an NFA are accepting depends on its language.

The Exponential Time Hypothesis (ETH)

Impagliazzo, Paturi and Zane [10, 11] observed that for many concepts similar to 3SAT the running time is exponential in the number n of variables. They investigated this topic and formulated what is now known as the Exponential Time Hypothesis: There is a constant $c > 0$ such that, every algorithm which solves 3SAT, has for infinitely many values of n the worst case time complexity of at least 2^{cn}. One defines $f \in \Omega'(g)$ iff there is a constant $c > 0$ such that for infinitely many n it holds that $f(n) > c \cdot g(n)$. Then one can obtain for many problems a conditional lower bound, that is, a lower bound implied by the Exponential Time Hypothesis where an absolute lower bound proof is unavailable. For example, for the unary NFA universality problem one gets the conditional lower bound of $2^{\Omega'(n^{1/3})}$.

In the present work, several lower bounds for the runtime to solve problems related to unary UFAs or NFAs are formulated under the assumption of the Exponential Time Hypothesis using Ω'-expressions. Furthermore, for constructing lower bounds, the following additional result of Impaglazzio, Paturi and Zane [10, 11] is important: Assuming the Exponential Time Hypothesis, there are constants $c, d > 1$ such that, for every algorithm solving 3SAT, there are infinitely many n such that for each of these n there is an instance with n variables, each occurring in the instance at most d times, such that the run time of the algorithm on this instance is at least c^n. Furthermore, one can show that one can choose $d = 3$ (at the expense of a perhaps smaller c, but still with $c > 1$).

The Prime Number Theorem

The prime number theorem says that ratio of the n-th prime number and $n \cdot \log n \cdot \log 2.71828\ldots$ converges to 1 where $2.71828\ldots$ is the Euler's number. A direct consequence of the prime number theorem is that there is a constant c such that, for each $n \geq 2$, there are $n/\log n$ prime numbers between n and cn.

In this paper, variations of these consequences are used in constructions to code 3SAT instances into NFAs and UFAs in order to show the hardness of decision problems. Often in proofs, the constructed NFA is in Chrobak Normal Form and consists of a stem of s states, where $s = 0$ is possible, and disjoint cycles $C_1, C_2, \ldots$ of lengths $p_1, p_2, \ldots$, with $n = s + p_1 + p_2 + \ldots$ being the overall number of states. In these cases, the states in cycle C_i will be considered to be ordered as 0-th, 1-st, ... states in the cycle, where the unary

alphabet takes the NFA from the current to the next state modulo p_i in the cycle C_i. If $s > 0$ the 0-th state in the cycle is the entry point into the cycle from the last state of the stem else each cycle has a start state which is its 0-th state. Assume that the stem has s states and an input of length t has been processed; if $s > t$ then the NFA is in the t-th state of the stem else the NFA is in the $(t - s \mod p_i)$-th state of the cycle C_i for some i.

Often (though not always) the p_i above would be either distinct prime numbers or distinct prime numbers times a common factor, the latter is in particular used for UFAs. This allows us to use Chinese Remainder Theorem to get that some possible combination of states is reachable in the different cycles for the same input word; in the case that all lengths are pairwise coprime, every combination can be reached.

3 The Nondeterministic Finite Automata Comparison Algorithm

The upper bound of the next result matches the lower bound $2^{\Omega'(n^{1/3})}$ of the timebound from the universality problem of unary NFAs [5] up to a factor $O((\log n)^{1/3})$ when comparing the logarithms of the corresponding run time bounds. A sketch of Tan's alternative proof [21] is in the appendix, as later results build on this method.

▶ **Proposition 1** (Fernau and Krebs [5]). *Given an m-variable 3-occur 3SAT instance, one can construct an $n = \Theta(m^3)$ sized NFA such that this NFA accepts all words over the unary alphabet iff the given instance is unsolvable. Thus unary NFA universality requires $2^{\Omega'(n^{1/3})}$ computation time provided that the Exponential Time Hypothesis holds.*

The upper bound $2^{((n \log n)^{1/3})}$ in the next result is only slightly larger than the lower bound $2^{\Omega'(\log n)^{1/3})}$. As the time used by an algorithm is at most $2^{O(\text{space used})}$, any further improvement in the space bound would also result in improvement in the time bound.

▶ **Theorem 2.** *Given two nondeterministic finite automata over the unary alphabet and letting n denote the maximum number of their states, one can decide whether the first NFA computes a subset of the second NFA in deterministic time $O(c^{(n \log n)^{1/3}})$ for a suitable constant $c > 1$. This timebound also applies directly to the comparison algorithm for equality and the algorithm for checking universality (all strings in the language). Furthermore, the algorithm can be adjusted such that the space used is $O((n \log n)^{1/3})$.*

Proof. Let n denote the maximum number of states of the two automata. Without loss of generality assume n is large enough so that the prime number theorem and other bounds needed below apply. One can transform (within polynomial time in the number of states) the nondeterministic finite automata into Chrobak Normal Form [2], where it consists of a stem of up to n^2 states, followed by parallel cycles which, together, use up to n states. Note that one can assume that the two stems have the same length. To see this, note that in Chrobak Normal Form, a stem can be made one longer by adding one state at the end of the stem and shifting the entry point into each cycle by one state. The new state in the stem would be accepting iff one of the prior entry points in the cycles was accepting. This is done repeatedly (at most $O(n^2)$ times) until the stems have the same length. The comparison of the behaviour on the stems of equal length before entering the cycles can be done by just comparing if the corresponding states at the same distance from the start are accepting/rejecting.

The comparison of the cycle part is therefore the difficult part. Thus, for the following, one assumes without loss of generality that the input NFAs are in Chrobak Normal Form, and do not have any stem. The NFAs thus consist only of disjoint cycles, each having one start state and the only nondeterminism is the choice of the start state, that is, the cycle to be used.

Note that a cycle C of length m in an NFA can be converted into a cycle C' of length w, where m divides w, by having the s-th state of C' as accepting if $(s \mod m)$-th state of C is accepting. This way, for appropriate value of w, one can combine several cycles whose lengths divide w into one cycle of length w. If one can get a set X of small number of w's (which pairwise have the same greatest common divisor (gcd) r), such that the lengths of all the cycles of the two NFAs divide at least one $w \in X$, then converting the two NFAs to have cycles only of lengths $w \in X$, allows for easier comparison of the two NFAs (these NFAs are called comparison normal form NFAs below).

Intuitively, lengths of only few (at most $(n/\log^2 n)^{1/3}$) cycles of the NFAs can have two large (at least $(n \log n)^{1/3}$) prime factors. The aim of P defined below is to collect all such prime factors along with the small primes. Set Q defined below collects the large primes which are not in P. Thus, P and Q together provide all the prime factors of the lengths of the cycles of the two NFAs.

Let $P = \{$prime number $p : p < (n \log n)^{1/3}$ or there exist a prime $q \geq (n \log n)^{1/3}$ such that one of the NFAs given in the input has a cycle with length divisible by $p \cdot q\}$.

Let $Q = \{$prime $p \leq n : p \notin P\}$.

Note that the number of primes smaller than $(n \log n)^{1/3}$ is at most $O((n/\log^2 n)^{1/3})$. Furthermore, the number of primes $p \geq (n \log n)^{1/3}$ such that the length of some cycle in one of the input NFAs is divisible by $p \cdot q$ for some prime $q \geq (n \log n)^{1/3}$ is at most $O((n/\log^2 n)^{1/3})$, as the cycles within each NFA are disjoint. Thus, the cardinality of P is at most $O((n/\log^2 n)^{1/3})$.

For each $p \in P$, let k_p be maximum number such that $p^{k_p} \leq n$. Note that p^{k_p} is the highest power of $p \in P$ which could divide the length of some cycle in the input NFAs. Similarly, q^2 is the highest power of $q \in Q$ which could divide length of any cycle in the input NFAs. Let r be the product of all p^{k_p}, $p \in P$. Note that r is in $O(n^{c' \cdot (n/\log^2 n)^{1/3}}) = O(2^{c' \cdot (n \log n)^{1/3}})$, for some constant c'. Thus, $r \leq c^{(n \log n)^{1/3}}$ for some constant $c > 1$.

Let $X = \{r \cdot q^2 : q \in Q\}$.

Note that lengths of all cycles in the two NFAs divide some $w \in X$. Moreover, the gcd of any two numbers in X is r. Now, for ease of comparing the two NFAs, one transforms each of these NFA into an equivalent "comparison normal form" NFA of size at most $r \cdot n^3$ as follows. For each $q \in Q$, the comparison normal form NFA has a cycle of length $r \cdot q^2$ such that the s-th state in this cycle is accepting iff there is a cycle of length p in the original automaton, where p divides $r \cdot q^2$ and $s \mod p$-th state in that cycle is accepting. Note that the comparison normal form NFA accepts the same language as the original NFA. The comparison normal form can be constructed in time $r \cdot Poly(n)$ by constructing each cycle separately, and comparing it with all cycles of length p in the original automaton, where p divides $r \cdot q^2$.

As the comparison normal form constructed is equivalent (for accepting language) to the original NFAs, it suffices to compare the two input NFAs in the comparison normal form, which is assumed below.

Now the first automaton recognises a subset of the set recognised by the second automaton iff for all $s < r$ one of the following two options holds:

(A) There is a $q \in Q$ such that in the second automaton, for all $t < q^2$, $(s + t \cdot r)$-th state in the cycle of length $r \cdot q^2$ are accepting;

(B) For every $q \in Q$ and for all $t < q^2$, if the $(s + t \cdot r)$-th state, in the cycle of length $r \cdot q^2$ is accepting in the first automaton then it is also accepting in the corresponding cycle of the second automaton.

This condition can be checked in time $r \cdot Poly(n)$: There are r possible values of s and for each such s, one has to check only $O(n^3)$ states, namely for each $q \in Q$, the $(s + t \cdot r)$-th state, where $t \in \{0, 1, \ldots, q^2 - 1\}$; note that $q^2 \leq n^2$.

For correctness, it is first shown that (A) and (B) are sufficient conditions. Let $s < r$ be given. If (A) is satisfied, then the second automaton, for all t, accepts strings of length $s + t \cdot r$, as for the given q, all these strings are accepted by the cycle of length $r \cdot q^2$.

If (B) is satisfied and the first NFA accepts a string of length $s+t\cdot r$, then there is a $q \in Q$ such that, in the cycle of length $r \cdot q^2$, the $(s + t \cdot r) \mod (r \cdot q^2)$-th state is accepting. From the condition it follows that in the second automaton, in the corresponding cycle, $(s + t \cdot r) \mod (r \cdot q^2)$-th state is accepting, and therefore it also accepts the corresponding string.

For the converse, assume that the following condition (C) holds: For every $q \in Q$ there exists a t_q such that the $(s + t_q \cdot r)$-th state is rejecting in the cycle of length $r \cdot q^2$ in the second automaton and furthermore, for one q, the $(s + t_q \cdot r)$-th state is accepting in the cycle of length $r \cdot q^2$ in the first automaton. So (C) is true iff both (A) and (B) are false. Now there is an s' such that, after processing a string of length s', the first automaton can be at $(s + t_q \cdot r)$-th state for the cycle of length $r \cdot q^2$, for each $q \in Q$. It follows that the first automaton accepts a string of length s' while the second automaton rejects it.

For the space-bounded variant of the algorithm, the algorithm cannot bring the automaton into a normal form, as that cannot be stored within the space allowed. The comparison algorithm therefore has the translation into the above used normal form more implicit. One does the following:

First one computes r. Then one needs constantly many variables bounded by $2n^4 \cdot r$, these variables can be stored in $O(\log r) = O((n \log n)^{1/3})$ space.

Second, for all cycles in the first automaton and all numbers $s \leq n^4 \cdot r$ such that a string of length s is accepted by the current cycle of some length p in the first automaton, one does the following: If p divides r then let $q = 1$ else let q be the unique prime such that p divides rq^2. Note that $q \leq n$. Now one checks if there is a number q' such that either $q' = 1$ or q' is a prime $\leq n$ which does not divide r and for each $\ell = 0, 1, \ldots, n^2$ there is a cycle of a length dividing rq'^2 in the second automaton which accepts the string of length $s + \ell r q^2$.

The language recognised by the first automaton is now a subset of the language recognised by the second automaton iff all the above tests in the algorithm have a positive answer. Note that r is chosen such that $\log c^{(n \log n)^{1/3}}$ and $\log r$ have the same order of magnitude and thus $O(\log c^{(n \log n)^{1/3}}) = O(\log(r \cdot n^4))$. Therefore, a real improvement of the space usage would also give an improvement of the computation time. ◀

4 Unambiguous Finite Automata and their Algorithmic Properties

Recall that an unambiguous automaton (UFA) satisfies that for every input word, there is either exactly one accepting run or none. On one hand, these are more complicated to handle than nondeterministic finite automata so that the union of n n-state automata cannot be done with n^2 states. On the other hand, they still, at least for unary alphabets, have good algorithmic properties with respect to regular operations (union, intersection, complementation, formation of Kleene star) and comparison (subset and equality).

▶ **Theorem 3.** *A UFA with up to n states has a complement with $n^{\log(n)+O(1)}$ states which can be computed in quasipolynomial time from the original automaton.*

Proof. Assume without loss of generality that the automaton is in Chrobak Normal Form. Furthermore, as inverting the states on the stem is trivial, it can be assumed without loss of generality, for an easier notation, that the given UFA consists just of m disjoint cycles $C_0, C_1, \ldots, C_{m-1}$ for some m, each having exactly one start state (denoted to be the 0-th state of the cycle).

Intuitively, the idea is to output a UFA using a recursive algorithm. At the start of the recursion, the aim is to output a UFA (without any stem) which accepts exactly the strings in the complement which have lengths $k \mod d$, with $k = 0, d = 1$. At each step of the recursion, with parameters k, d, either the algorithm

(a) returns a UFA (without any stem) which accepts exactly the strings in the complement which have lengths $k \mod d$ or

(b) makes recursive calls to obtain UFAs (without any stem) for accepting exactly the strings in the complement with lengths $(k + d \cdot s) \mod (d \cdot \ell)$, for some value of ℓ and s being $0, 1, \ldots, \ell - 1$. As the above UFAs would be without any stem accepting disjoint languages, the union of these UFAs will give a UFA for accepting exactly the strings in the complement which have lengths $k \mod d$.

Though, the algorithm is presented as a recursive algorithm, one can also view the solution as a tree, where the root of the tree has parameters $(k = 0, d = 1)$. Any node of the tree is either a leaf (i.e., it gives a UFA, without any stem, for accepting exactly the strings in the complement which have lengths $k \mod d$), or has ℓ children, for some ℓ, with parameters $(k + sd, d \cdot \ell)$, for s being $0, 1, \ldots, \ell - 1$ respectively in the ℓ children. The UFA for the complement thus will be the union of the UFAs at the leaves. Note that for any two leaves with parameters k', d' and k'', d'' there is no length m with m being same as both $k' \mod d'$ and $k'' \mod d''$. Thus, the above tree is also called a tree of different modulo residua.

Now the formal recursive algorithm is presented. Initially the algorithm is called with parameters $k = 0$ and $d = 1$.

Function UFAcomplement(k, d)

1. If there is a cycle C_i in the input UFA such that all strings of length $k + sd$ with $s < |C_i|$ are accepted by this cycle, then return to the calling instance of the recursion a UFA for emptyset (as the input UFA accepts all strings of length $k \mod d$, UFA for the complement needs to reject all strings of length $k \mod d$).
2. If there is no cycle C_j accepting any string of length $k + sd$ with $s < |C_j|$, then return one cycle of length d for which the k-th state is accepting and all other states are rejecting (as the given input UFA rejects all strings of length $k \mod d$, the UFA for complement needs to accept all such strings).
3. Otherwise there is a cycle C_h which accepts some but not all strings of the length $k \mod d$. The algorithm computes now the least common multiple $d' = lcm(d, |C_h|)$ and makes, for $s = 0, 1, \ldots, d'/d - 1$ a recursive call with the parameters $(k + ds, d')$. Return the union of the answers obtained from the recursive calls.

End of function UFAcomplement(k, d)

The algorithm clearly terminates, as when d is the multiple of all the cycle lengths and k is a number between 0 and $d - 1$, then every cycle C_i has the property that it either accepts all strings of length $k + sd$ or rejects all strings of length $k + sd$. However, the following claim shows that the value of d is much smaller at the termination step.

▷ Claim 4. Value of d at the termination step is at most $n \cdot (n/2) \cdot (n/2^2) \ldots \leq n^{0.5 \log n + c}$, for some constant c.

To see the claim, consider any branch of the recursive descent, with the values of (k, d) in the recursive calls being (k_0, d_0) (at the root), (k_1, d_1), ..., where when the values were (k_i, d_i), then cycle C_{e_i} is chosen in step 3 (except at the last level which terminates in step 1 or step 2). For i not being the last level of the recursive descent, following properties hold:

(i) $d_{i+1} = lcm(d_i, |C_{e_i}|)$. In particular, d_i divides d_{i+1}.

(ii) $k_{i+1} = k_i + s_i d_i$, for some s_i.

(iii) C_{e_i} accepts some but not all strings of length $k_i \mod d_i$.

(iv) $|C_{e_i}|$ does not divide d_i but divides d_{i+1}.

Thus, for any levels g, h not being the last level of the recursive descent with $g < h$, using (i) and (ii) repeatedly, $k_h = k_g + s_{g+1}d_{g+1} + \ldots + s_{h-1}d_{h-1} = k_g + s'_h d_g$, for some s'_h, and d_g divides d_h. Using (iii) both C_{e_g} and C_{e_h} accept some words of length $k_g \mod d_g$. Thus, by UFA property of the input NFA, there is a common factor $b > 1$ of $|C_{e_g}|$ and $|C_{e_h}|$ which does not divide d_g. Note that b divides d_{g+1} as $|C_{e_g}|$ divides d_{g+1} (by (iv)). Thus, there is an extra common factor greater than 1 between d_{g+1} and $|C_{e_h}|$ compared to d_g and $|C_{e_h}|$, for each $g < h$. Thus common factor between d_g and $|C_{e_h}|$ is at least 2^g. It follows that d_{h+1}/d_h is at most $n/2^h$.

Thus, the number of levels is at most $\log n$ and the value of d is at most $n \cdot (n/2) \cdot (n/2^2) \ldots \leq n^{0.5 \log n + c}$, for some constant c, where one can safely assume $c \leq 5$. This proves the claim.

Also, it is easy to see by induction that the automaton generated by the algorithm is a UFA, as UFAcomplement(k, d) either returns a UFA in steps 1 or 2, or combines the UFAs generated by recursive calls in step 3, sets accepted by which are disjoint as they only accept strings of length $k + ds \mod d'$, for different values of s.

The automaton is the union of cycles up to length $n^{\log n/2 + c}$ and in some kind of post-processing, one can unify distinct cycle of the same length d, with k_1-th, ... k_s-th states as accepting into a single cycle of length d which has the k_1-th, ..., k_s-th states as accepting states. After this post-processing, there are at most $n^{(\log n)/2 + c}$ cycles and thus the overall size is at most $n^{\log n + 2c}$. ◀

▶ **Theorem 5.**

(a) *One can decide in* ***LOGSPACE*** *whether an UFA in Chrobak Normal Form accepts all words. Without the* ***LOGSPACE*** *constraint, the running time is quasilinear, that is, of the form* $O(n \log^2(n))$.

(b) *Furthermore, one can decide in* ***LOGSPACE*** *whether an UFA* U_1 *in Chrobak Normal Form accepts a subset of the language accepted by another UFA* U_2 *in Chrobak Normal Form.*

Proof.

(a) First check if the states of the stem are all accepting, which can clearly be done in **LOGSPACE** by automata walk-through - pointers needed $O(\log n)$ memory to track positions in the UFA. Then one walks through each cycle and counts the number i_k of accepting states and the length j_k of the cycle. Now the UFA accepts all words, that is, is universal iff $\sum_k i_k/j_k = 1$. The proof starts by initially showing how to do this without being careful about space, but later it is shown how the computation can be modified to be done in **LOGSPACE**.

As the computation with rational numbers might be prone to rounding, one first normalises to one common denominator, namely $p = \prod_k j_k$ and furthermore computes $s = \sum_k i_k \cdot \prod_{h \neq k} j_k$. Now the above equality $\sum_k i_k/j_k = 1$ holds iff $s = p$.

The values of s and p can be computed iteratively by the following algorithm, note that there are at most n cycles and each time a cycle is processed, the corresponding values i_k and j_k can be established by an automata walk-through. So the loop is as follows:

1. Initialise $s = 0$ and $p = 1$;
2. For each k do find i_k and j_k; update $s = (s \cdot j_k) + i_k \cdot p$; $p = p \cdot j_k$ endfor;
3. if $s = p$ then accept else reject.

In this algorithm, only the variables p and s need more space than $O(\log n)$; the other variables are all pointers or numbers between 0 and n which can be stored in $O(\log n)$ space.

The values of s and p are at most $n^{2(n^{1/2})}$. To see this note that in Chrobak Normal Form, the cycle lengths can be assumed to be different (as same length cycles can be combined). Thus, there are at most $2n^{1/2}$ cycles, as the sum of their lengths is at most n implying that at most $n^{1/2}$ cycles have at least length $n^{1/2}$ and, furthermore, there are at most $n^{1/2}$ cycles shorter than $n^{1/2}$ due to different cycles having different length. Thus, instead of doing the above algorithm once with exact numbers, one computes in time $O(n \log n)$ the first $5 \cdot n^{1/2} + 2$ primes out of which 80% are above $n^{1/2}$ so that their product is above the maximum values s and p can take. As their product is larger than the upper bound $n^{1+2(n^{1/2})}$ of s and p, one then accepts iff all computations modulo each such prime q result in $s = p$ modulo q; this condition is, by the Chinese remainder theorem, equivalent to $s = p$ without taking any remainders. So the modified algorithm would be as follows.

1. Let $q = 2$; $\ell = 1$;
2. Initialise $s = 0$ and $p = 1$ (both are kept modulo q)
3. For each k do find i_k and j_k by transversal of the corresponding cycle; update (both computations modulo q) $s = (s \cdot j_k) + (i_k \cdot p)$; $p = p \cdot j_k$ endfor;
4. if $s \neq p$ (modulo q) then reject;
5. Let $\ell = \ell + 1$ and replace q by the next prime using a fast primality test and exhaustive search;
6. If $\ell < 5n^{1/2} + 2$ then goto step 2.

The automata transversal of each of these $O(n^{1/2})$ loops needs at most time $O(n \log n)$ as one transverses each of the cycles of the UFA to determine the corresponding i_k and j_k, the cycles are disjoint and have together at most n states. So the overall running time is $O(n^{3/2} \log n)$. If one would use more space, about $O(n^{1/2} \log n)$, then one can avoid the repeated computation of i_k, j_k by automata walk-throughs and gets the upper bound on the computation time of $O(n \log^2(n))$.

For the space usage, the computations modulo q need only $O(\log q)$ space which is then $O(\log n)$ space, as the first $5n^{1/2} + 2$ primes q and the storage of variables modulo q is all of $O(\log n)$. Primality tests can be done in $O(\log n)$ space for the usual way of doing it – checking all divisors up to the square root of the number.

(b) For this, note that basically the same idea as in (a) can be used to check if a UFA accepts all unary strings in sets vw^* for some v, w of length up to $2n$ with a slight modification of the UFA walk-throughs.

One partitions the words accepted by U_1 into two groups:

(i) A finite set X_1 of strings of length at most n (where n is the size of the UFA) and

(ii) A set X_2 consisting of subsets of the form vw^*, where v, w are unary strings with $n < |v| \leq 2n$ and $|w| \leq n$.

The strings in group (ii) above are from the cycles in U_1, by considering each accepting state in a cycle separately, and taking $|v|$ as the length of the smallest string leading to the state and $|w|$ as the length of the corresponding cycle.

Strings in group (i) can easily be checked using a walk-through of U_2, where if there is a branching into the cycles, one can do a depth first search.

For strings in group (ii), each set of form vw^*, where $n < |v| \leq 2n$ and $|w| \leq n$, is checked separately. As $|v| >$ the length of the stem part of U_2, one can first modify the cycle part of U_2 to always start in a state which is reached after $|v|$ steps, and ignore the stem part. This

would basically mean that one needs to check if w^* is accepted in the modified U_2 (denote this modified U_2 as U_2'). For space constraints, note that one does not need to write down U_2', but just need to know the length by which the starting state of each cycle is shifted (which is the difference between $|v|$ and the length of the stem part of U_2). Now, for checking whether w^* is accepted by U_2', consider a further modified U_2'' formed as follows: for each cycle C in U_2' with length r and states $s_0, s_1, \ldots, s_{r-1}$ (s_0 being starting state, and transitions on unary input being from s_i to s_{i+1}, where $i+1$ is taken mod r) consider a cycle C' in U_2'' with states $s_0', s_1', \ldots, s_{r-1}'$ (s_0' being starting state, and transitions on unary input being from s_i' to s_{i+1}', where $i+1$ is taken mod r) where s_i' is an accepting state iff $s_{|v|+i\cdot|w| \mod r}$ was an accepting state in C. This new UFA U_2'' also has at most n states, the new length of each cycle is still the same as the lengths of the old cycles and the number of cycles do not increase. Now, similar to part (a), one just needs to check if U_2'' is accepting all unary strings. Here again note that one doesn't need to write down U_2'' fully, but just needs to check, for each cycle, its length and the number of accepting states, which can be done in **LOGSPACE**. ◀

Converting an UFA into Chrobak Normal Form is in **P** and **LOGSPACE** $\subseteq$ **P**, thus the following corollary holds.

▶ **Corollary 6** (Stearns and Hunt [20]). *One can decide the universality problem and the inclusion problem for two n-state UFAs in polynomial time.*

▶ Remark 7. One can improve Corollary 6 to computations in **NLOGSPACE** and, by Savitch's Theorem [19], in **DSPACE**$((\log n)^2)$. Details are in the appendix.

▶ Remark 8. For an UFA (of size n) for a language L over the unary alphabet, the language L^* can be recognised even by a DFA of quadratic size in n [1].

Thus if one allows in regular languages Kleene star, Kleene plus and the Boolean set-theoretic operations (but no concatenation), then the output of constant-size expressions, with parameters being given by languages accepted by n-state UFAs, can be recognised by UFAs of quasipolynomial size. Furthermore, Boolean-valued constant-sized quantifier-free formula with same type of parameters and comparing such subexpressions by $=$, $\subseteq$ and $\neq$ can be evaluated in quasipolynomial time.

Furthermore, one can show, as **NLOGSPACE** $\subseteq$ **POLYLOGSPACE**, that the above mentioned operations can all be computed by **POLYLOGSPACE** algorithms. This is not true for the concatenation, as the next Theorem 9 shows that there is an exponential-type blow-up – therefore **POLYLOGSPACE** is not enough to count the number of output symbols. Here some more details for the regular operations different from concatenation.

For the complementation, the handling of the stem is standard. The algorithm in Theorem 3 above has mainly two running variables for the recursive descent: d and k. Both take at most the value $n^{\log n/2+c}$ for some constant c and therefore can be written with $O(\log^2 n)$ bits. Furthermore, during recursion, one has to archive the old values before branching off, thus the algorithm archives $(d_{h'}, k_{h'}, e_{h'}, s_{h'})$ from the algorithm in Theorem 3 for each level h' and this can be done with $O(\log^3 n)$ space. Furthermore, note that the transformation into Chrobak Normal Form also takes just $O(\log^2 n)$ space due to Savitch's Theorem: $O(\log n)$ nondeterministic space is contained in $O(\log^2 n)$ deterministic space. In particular the algorithm is a **POLYLOGSPACE** algorithm. The above space bound can be improved somewhat by noting the following: instead of archiving the full tuples $(d_{h'}, k_{h'}, e_{h'}, s_{h'})$, one could archive the index information $e_{h'}, s_{h'}$ when going from level h' to $h'+1$, along with the quotient $d_{h'+1}/d_{h'}$. When returning one level down, one computes $d_{h-1} = d_h/(d_h/d_{h-1})$ where d_h/d_{h-1} was archived for $h-1$ and furthermore, one computes $k_{h-1} = k_h - s_{h-1} \cdot d_{h-1}$. With these modifications, the algorithm runs in $O(\log^2 n)$ space.

The intersection of two automata can just be done by computing the product automaton. This is known to be doable in **LOGSPACE**.

The union is the complement of the intersection of the complements and, by the above algorithms, can be done in **POLYLOGSPACE**.

Yu, Zhuang and Salomaa [22] showed that the Kleene star of a unary language given by an n-state NFA can be computed by a DFA of size $(n-1)^2+1$; thus one can search using an **NLOGSPACE** algorithm for the first $m \geq 1$ such that, for $k = 0, 1, \ldots, (n+1)^4$, every word of length n^2+k are accepted iff the word of length n^2+m+k is accepted and this length is then the period which starts latest at length n^2. This allows to output a DFA of size $O(n^2)$ which consists of a stem of length n^2 and a period of length m, where the **NLOGSPACE** algorithm allows for each of the states involved to check whether it is accepting.

▶ **Theorem 9.** *There is an exponential-type blow-up for UFA sizes when recognising the concatenation of unary languages; the concatenation of two languages given by n-state UFAs requires, in the worst case, an UFA with $2^{\Omega(n^{1/6})}$ states.*

Proof. Let m be a numeric parameter and let $p_0, \ldots, p_{k-1}$ be the first $k = m/\log m$ primes of size at least m; note that $m > k+4$ for all sufficiently large m. These primes are within $\Theta(m)$ by the prime number theorem. Now the UFA U to be constructed contains k cycles C_ℓ of length $p_\ell \cdot (k+3)$ for $\ell = 0, 1, \ldots, k-1$. The cycle C_ℓ has, for $h = 0, 1, \ldots, p_\ell - 2$, $(\ell + 2 + h \cdot (k+3))$-th state as accepting. There is one further cycle C' of length $k+3$ which has 0-th, 1-st and $k+2$-th states as accepting. Let L denote the language recognised by this UFA. The lengths of k consecutive unary strings not being accepted by the above UFA are exactly at lengths $r \cdot (k+3)+2, \ldots, r \cdot (k+3)+k+1$ where r is $p_\ell - 1$ modulo p_ℓ for $\ell = 0, 1, \ldots, k-1$, and this does not happen at any other lengths. Let H be the finite language which contains the words of length $0, 1, \ldots, k-1$ and no other words. Now $L \cdot H$ contains all words whose length does, for at least one ℓ, not have the remainder $k+1+(k+3)\cdot(p_\ell - 1)$ modulo $(k+3)\cdot p_\ell$. The complement of $L \cdot H$ is a periodic language which contains exactly those words which have, for all ℓ, the remainder $k+1+(k+3)\cdot(p_\ell-1)$ modulo $(k+3)\cdot p_\ell$. So every NFA or UFA recognising this language needs cycles of length at least $(k+3)\cdot p_0 \cdot p_1 \cdot \ldots \cdot p_{k-1}$. The length of this cycle is at least $\Theta(m^k \cdot (k+3))$ and any UFA recognising it needs at least the same number of states. Furthermore, the UFA U has n states with $n \in \Theta(m) \cdot \Theta(k^2) = \Theta(m^3/\log^2 m)$. It follows that $n \cdot \Theta(\log^2(m)) = \Theta(m^3)$ and, using $\Theta(\log n) = \Theta(\log m)$ as n, m are polynomially related, that $\Theta(n \cdot \log^2(n)) = \Theta(m^3)$. So $m \in \Theta((n \cdot \log^2 n)^{1/3})$. Now the concatenation of $L \cdot H$ has an UFA of size o to be determined and its complement, by the above, has an UFA of at least size $(m)^{m/\log m}$. Using that the complement of an UFA of size h can be done in size $h^{\log h + O(1)}$, one has that the logarithm of the corresponding sizes satisfies this: $\log^2 o \geq \Theta((\log m + O(1)) \cdot m/\log m) \geq \Theta(m)$ and $\log o \geq \Theta(m^{1/2})$. Now the input UFA is of size n with $m \in \Theta((n \log^2 n)^{1/3})$ and thus one has that $\log o \in \Omega(n^{1/6})$. So concatenating two languages given by n-state UFAs can require an UFA with $2^{\Omega(n^{1/6})}$ states. ◀

▶ Remark 10. The above result stands in contrast to the situation of NFAs. It is well-known that the concatenation of two n-state NFAs needs only $2n$ states. However, the above construction shows that forming the complement in the unary NFAs then blows up from $2n$ states to $2^{\Omega(n^{1/6})}$ states, giving an exponential-type lower bound for this operation; a direct construction leading to the bound $2^{\Omega((n \log n)^{1/2})}$ is known [15]. Furthermore, Pighizzini [16, 17] showed that the concatenation of two unary DFAs of size n can be realised by an DFA of size $O(n^2)$; actually, he gives an explicit formula on the size of the stem and the cycle which depends only on the size of the stems and the cycles in the two input automata. Pighizzini's

result allows for an implementation of the following concatenation algorithm [16, 17]: Convert the two UFAs into DFAs and then apply the algorithm to make the concatenation of DFAs; this gives the upper bound of $2^{O((n \log^2 n)^{1/3})}$ for the size of the concatenation UFA.

▶ Remark 11. The following facts are known about the complexity of operations with UFAs.

Holzer and Kutrib [7] showed that the intersection of two UFAs is just given by the n^2 state sized product automaton of the two n-state UFAs which preserves the property of being UFAs.

The bound $2n$ for the disjoint union by Jirásková and Okhotin [14] are obtained by the simple union of the two UFAs (this might give multiple start states, which is allowed for UFAs); note that as the languages are disjoint, on each word in the union, one can reach as accepting state only in one of the UFAs and there, by assumption, the way into the accepting state is unique. The lower bound $2n - 4$ for the disjoint union can be obtained by two even length cycles differing by length 2; one cycle has the accepting states at some of the odd numbered states and the other one at some of the even numbered states.

For the general union of two languages L, H given by n-state UFAs, consider the formula $L \cup (H - L)$ where $H - L$ is equal to the intersection of H and the complement of L which gives the upper bound $n \cdot c(n) + n$ with $c(n)$ being the bound for complementation.

The symmetric difference of two languages L, H is the disjoint union of $L - H$ and $H - L$, thus an upper bound is $2n \cdot c(n)$. For its lower bound, note that if L is given and H is the set of all strings, then the symmetric difference $(L - H) \cup (H - L)$ is just the complement of L.

Finite formulas refers to a formula with several input automata combining the input n-state UFAs with regular operations to a new UFA. Subsequent results provide the lower bound of $O(2^{\Omega'(n^{1/3})})$ for the time complexity to evaluate formulas. This almost matches the upper bound provided by transforming the UFAs into DFAs and exploiting the polynomial bounds on size increase of DFAs over the unary alphabet for regular operations.

In summary: All standard regular operations except concatenation have polynomial or quasipolynomial size-increase but concatenation has exponential-type size-increase.

Next one considers the evaluation of constant-sized formulas using UFAs as input. In the following, H_1, H_2, L are sets of words given by n-state UFAs and K is a finite language.

▶ **Theorem 12.** *Assuming the Exponential Time Hypothesis, it needs time* $2^{\Omega'((n \log n)^{1/3})}$ *to evaluate the truth of the formula* $(H_1 \cap H_2) \cdot K = L$ *where* H_1, H_2, K *are given by at most* n*-state UFAs and* L *is the set of all words.*

Proof. Consider a 3SAT formula with clauses $c'_1, c'_2, \ldots, c'_m$, where each variable appears in at most 3 clauses. Divide the clauses into $r = \lceil m/\log m \rceil$ disjoint groups of $\log m$ clauses each (where the rounded down value of $\log m$ is used). Group G_i has the clauses $c'_{(i-1)\cdot(\log m)+1}, \ldots, c'_{i\cdot(\log m)}$ (the last group G_{r-1} may have lesser number of clauses due to m not being divisible by $\log m$).

If a variable, say x, appears in different groups of clauses, then one can rename x in different groups to $x', x'', \ldots$ and add equality clauses $(x' = x''), \ldots$. Thus, by adding some additional equality clauses, it can be assumed that no variable appears in two different group of clauses. Note that there can be at most $O(m)$ equality clauses.

So, now the SAT formula has the clauses $c_1, c_2, \ldots, c_m$ (which have at most 3 literals each) and the equality clauses $c_{m+1}, c_{m+2}, \ldots, c_{m'}$, where the clauses $c_1, c_2, \ldots, c_m$ are divided into groups $G_0, G_1, \ldots, G_{r-1}$ containing at most $\log m$ clauses each, and any variable appears in clauses from at most one group, and perhaps in equality clauses.

As each group contains at most $\log m$ clauses, and thus at most $3 \log m$ variables, there are at most $8m$ possible truth assignments to variables appearing in clauses of any group. Below, k-th truth assignment (starting from $k = 0$) to variables assigned to p_i assumes some

Table 4 Accepting and Rejecting States.

	$k \cdot (2m' + 2) + 2j$-th state in C_i / C'_i	$k \cdot (2m' + 2) + 2j + 1$-th state in C_i / C'_i
$j = 0$, $i = 0$	accepting	accepting
$0 < j \leq m$ and all variables of c_j belong to G_i	accepting iff c_j not satisfied by the k-th truth assignment to variables assigned to p_i	accepting iff c_j not satisfied by the k-th truth assignment to variables assigned to p_i
This row is applicable only for C_i where $m < j \leq m'$ and c_j is $(x = y)$ and x is assigned to p_i	accepting iff truth value assigned to x is true in the k-th truth assignment to the variables assigned to p_i	accepting iff truth value assigned to x is false in the k-th truth assignment to the variables assigned to p_i
This row is applicable only for C'_i where $m < j \leq m'$ and c_j is $(x = y)$ and y is assigned to p_i	accepting iff truth value assigned to y is false in the k-th truth assignment to the variables assigned to p_i	accepting iff truth value assigned to y is true in the k-th truth assignment to the variables assigned to p_i
All other cases	not accepting	not accepting

ordering among the truth assignments where if the number of truth assignments is less than k, then k-th truth assignment is assumed to be the 0-th truth assignment (the latter part is just for ease of writing the proof).

Consider r distinct primes $p_0, p_1, \ldots p_{r-1}$, each greater than $8m$ but below some constant c times m. Note that, for large enough constant c, there exist such distinct primes by the prime number theorem.

Assign variables / clauses appearing in group G_i to prime p_i. Now UFA for H_1 consists of the r cycles $C_0, C_1, \ldots, C_{r-1}$ and UFA for H_2 consists of the r cycles $C'_0, C'_1, \ldots, C'_{r-1}$. The cycles C_i and C'_i are of length $2(m' + 1) \cdot p_i$. For $k < p_i$, and $j < m'$, the $k \cdot (2m' + 2) + 2j$-th and $k \cdot (2m' + 2) + 2j + 1$-th states in cycle C_i and C'_i are accepting or non-accepting as given in Table 4. Intuitively, for c_j assigned to p_i, C_i and C'_i will test for satisfaction (3rd row in the table). If $c_j = (x = y)$, and x is assigned to p_i and y to $p_{i'}$, then 4th and 5th rows in the table for C_i and $C'_{i'}$ respectively will check for consistency in the assignment to the variables x and y. Note that in Table 4 for the entries of the cycles C_i and C'_i, the value i can be considered as constant and the value j is the running variable going over all clauses.

Note that the above automatons are unambiguous, as for $j = 0$, only cycle C_0, C'_0 could accept; for $1 \leq j \leq m$, only C_i, C'_i such that c_j is assigned to p_i can accept; for $m < j \leq m'$, if c_j is $(x = y)$ with x and y assigned to p_i and $p_{i'}$ respectively, only C_i and $C'_{i'}$ respectively can accept. All strings with length being 0 or 1 modulo $(2m' + 2)$ are in $H_1 \cap H_2$ as defined in the table.

Now, if the 3SAT formula is satisfiable, then consider some satisfying truth assignment to the variables, say it is the k_i-th truth assignment to variables assigned to p_i. Then, for any s such that for $i < r$, $k_i = s \mod p_i$, $H_1 \cap H_2$ will not contain strings of length $s(2m' + 2) + 2j$ and $s(2m' + 2) + 2j + 1$ for $1 \leq j \leq m'$: (a) if $1 \leq j \leq m$, clause c_j is satisfied and thus these strings are not accepted by H_1 and H_2 (see 3rd row in the table for definition of C_i and C'_i, for c_j being assigned to p_i); (b) if $m < j \leq m'$ and $c_j = (x = y)$, where x is assigned

to p_i and y to $p_{i'}$, then as the variable assignments are consistent again these strings are accepted by only one of H_1 and H_2 (see 4th and 5th rows in the table for definition of C_i and $C'_{i'}$ respectively). Thus $H_1 \cap H_2$ misses $2m'$ consecutive strings.

On the other hand if $H_1 \cap H_2$ misses $2m'$ consecutive strings, then it must be strings of length $s(2m'+2)+j$ for $2 \leq j \leq 2m'+1$, for some value of s (as all strings with lengths being 0 or 1 modulo $2m'+2$ are in $H_1 \cap H_2$). Then let $k_i = s \mod p_i$. Then, for the k_i-th assignment of truth values to the variables assigned to p_i, it must be the case that all the clauses c_j assigned to p_i are satisfied (otherwise by 3rd row in the table for definition of C_i, C'_i, H_1 and H_2 will contain $s(2m'+2)+2j$ and $s(2m'+2)+2j+1$). Furthermore, the equality clauses are satisfied, as if some equality clause $c_j = (x = y)$ is not satisfied then, if x is assigned to p_i and y is assigned to $p_{i'}$, then by the 4th and 5th rows in the table for definitions of C_i and $C_{i'}$ respectively, one of $s(2m'+2)+2j$ and $s(2m'+2)+2j+1$ length strings is in both $H_1 \cap H_2$ depending on the truth assignment to x in the k_i-th and y in the $k_{i'}$-th truth assignments to the variables assigned to p_i and $p_{i'}$ respectively.

Thus, $H_1 \cap H_2$ misses out on $2m'$ consecutive strings iff the given 3SAT formula is not satisfiable. Taking K to be set of strings of length $1, 2, \ldots, 2m'-1$, gives us that $(H_1 \cap H_2) \cdot K$ is universal iff 3SAT formula is not satisfiable.

The size n of the UFAs for H_1, H_2, K is bounded by $(2m'+2) \cdot (\sum_i p_i) = \Theta(m^3/\log m)$. Thus, $n \log n = \Theta(m^3)$ or $m = \Theta((n \log n)^{1/3})$. The theorem now holds assuming ETH. ◀

Note that Okhotin [15] provides an upper bound by converting the UFAs into DFAs and then carrying out the operations with DFAs. These operations run in time polynomial in the size of the DFAs constructed. While the size lower bound for concatenation of two n-state UFAs is just $2^{\Omega(n^{1/6})}$, the following conditional bound on the computational complexity of finding an UFA for the concatenation is $2^{\Omega'(n^{1/4})}$ when using the Exponential Time Hypothesis – for details see the appendix.

▶ **Theorem 13.** *Under the assumption of the Exponential Time Hypothesis, one needs at least $2^{\Omega'(n^{1/4})}$ time to compute an UFA for the language of the concatenation of the languages of two given n-state UFAs in worst case.*

▶ Remark 14. The above result also proves that for deciding whether the concatenation of two languages given by n-state UFAs is universal, the Exponential Time Hypothesis implies a $2^{\Omega'(n^{1/4})}$ lower bound. The upper bound of this is slightly better than the DFA-conversion bound $2^{O((n \log^2 n)^{1/3})}$ of Okhotin [15]: Theorem 2 proves that the universality of an NFA can be checked in time $2^{O((n \log n)^{1/3})}$ and as the concatenation of two n-state UFAs can be written as an $2n$-state NFA whose universality can be checked with the same upper bound.

5 Conclusion

The main results of the present work are the following ones: (1) There is an $2^{O((n \log n)^{1/3})}$ time algorithm for comparing NFAs with respect to $=$ and $\subseteq$; (2) given n-state UFAs, the size of UFAs for their union and complement is at most $O(n^{\log n + O(1)})$ states; (3) the concatenation of two n-state UFAs needs, in the worst case, at least $2^{\Omega(n^{1/6})}$ states. Furthermore, conditional lower bounds and upper bounds for the time and space complexities of operations and decision problems on UFAs are provided. The main question remaining unresolved is to close the gap between the lower bound $2^{\Omega(n^{1/6}))}$ and Okhotin's upper bound $2^{O((n \log^2 n)^{1/3})}$ for concatenating two n-state UFAs. Several related results can be found in the longer technical report version of this paper on `https://arxiv.org/abs/2302.06435`.

References

1. Kristína Čevorová. Kleene star on unary regular languages. In *Descriptional Complexity of Formal Systems: 15th International Workshop, DCFS 2013, London, ON, Canada, July 22-25, 2013. Proceedings 15*, volume 8031 of *LNCS*, pages 277–288. Springer, 2013.
2. Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47:149–158, 1986.
3. Thomas Colcombet. Unambiguity in automata theory. In *Seventeenth International Workshop on Descriptional Complexity of Formal Systems*, volume 9118 of *LNCS*, pages 3–18. Springer, 2015.
4. Maciej Dębski. State complexity of complementing unambiguous automata, Master's thesis, University of Warsaw, 2017.
5. Henning Fernau and Andreas Krebs. Problems on finite automata and the exponential time hypothesis. *Algorithms*, 10(1):24, 2017.
6. Mika Göös, Stefan Kiefer, and Weiqiang Yuan. Lower bounds for unambiguous automata via communication complexity. In *Forty Ninth International Colloquium on Automata, Languages and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 126:1–13, 2022. See also the technical report on `arXiv:2109.09155`.
7. Markus Holzer and Martin Kutrib. Unary language operations and their nondeterministic state complexity. In *International Conference on Developments in Language Theory*, volume 2540 of *LNCS*, pages 162–172. Springer, 2002.
8. Markus Holzer and Martin Kutrib. Descriptional and computational complexity of finite automata – A survey. *Information and Computation*, 209(3):456–470, 2011.
9. John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 3rd edition, 2007.
10. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
11. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
12. Emil Indzhev and Stefan Kiefer. On complementing unambiguous automata and graphs with many cliques and cocliques. *Information Processing Letters*, 177:106270:1–5, 2022.
13. Jozef Jirásek Jr, Galina Jirásková, and Juraj Šebej. Operations on unambiguous finite automata. *International Journal of Foundations of Computer Science*, 29(05):861–876, 2018.
14. Galina Jirásková and Alexander Okhotin. State complexity of unambiguous operations on deterministic finite automata. In *Twentieth International Conference on Descriptional Complexity of Formal Systems*, volume 10952 of *LNCS*, pages 188–199. Springer, 2018.
15. Alexander Okhotin. Unambiguous finite automata over a unary alphabet. *Information and Computation*, 212:15–36, 2012.
16. Giovanni Pighizzini. Unary language concatenation and its state complexity. In *Implementation and Application of Automata: 5th International Conference, CIAA 2000 London, Ontario, Canada, July 24–25, 2000 Revised Papers 5*, volume 2088 of *LNCS*, pages 252–262. Springer, 2001.
17. Giovanni Pighizzini and Jeffrey Shallit. Unary language operations, state complexity and Jacobsthal's function. *International Journal of Foundations of Computer Science*, 13(01):145–159, 2002.
18. Michael Raskin. A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton. In *Fortyfifth International Colloquium on Automata, Languages and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 138:1–11, 2018.
19. Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.
20. 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.

21 Christopher Tan. Characteristics and computation of minimal automata, UROP, National University of Singapore, 2022.
22 Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994.

A Appendix of Outcommented Proofs

▶ **Proposition 1** (Fernau and Krebs [5], Tan [21]). *Given an m-variable 3-occur 3SAT instance, one can construct an $n = \Theta(m^3)$ sized NFA such that this NFA accepts all words over the unary alphabet iff the given instance is unsolvable. Thus unary NFA universality requires $2^{\Omega'(n^{1/3})}$ computation time provided that the Exponential Time Hypothesis holds.*

Proof. Suppose an m-variable 3SAT instance with at most $3m$ clauses, where each variable occurs at most three times, is given. Without loss of generality assume $m \geq 8$ and $\log m$ is a whole number. Let the clauses be $u_1, u_2, \ldots, u_{m'}$ (where $m' \leq 3m$) and the variables be $x_1, \ldots, x_m$.

Let $r = \lfloor \frac{\log m}{3} \rfloor$, and $s = \lceil m'/r \rceil$. Consider the primes $p_0, p_1, \ldots, p_{s-1}$, where $8m \leq p_i \leq c'm$, for some constant c'. Note that by the prime number theorem there exists such a constant c'.

Assign to each prime p_i, the clauses $u_{(i-1)\cdot r+1}, \ldots, u_{i\cdot r}$. Intuitively, for each $i < s$, there will be a cycle of length p_i which will explore all possible truth assignments to variables in the clauses assigned to p_i, and check whether they satisfy the corresponding clauses assigned. Consistency of truth assignments to variables across clauses assigned to different primes p_i and p_j will be checked using a cycle of size $p_i \cdot p_j$. Further details can be found below.

Note that r clauses have at most $3r$ literals, and thus the number of possible truth assignments to these literals is at most 2^{3r} with $2^{3r} \leq m \leq p_i$. Order these assignments in some way so that one can say k-th truth assignment etc.

(1) For each $i < s$, form a cycle of length p_i. The k-th state (for $k < 2^{3r}$) in this cycle is rejecting iff the k-th truth assignment to the $3r$ literals in the clauses assigned to p_i satisfy all the clauses assigned to p_i. Note that if $k \geq 2^{3r}$, then the k-th state is accepting.

(2) For each pair i, j such that clauses assigned to p_i and p_j have a common variable, form a cycle of length $p_i \cdot p_j$. The k-th state in this cycle is accepting iff the $(k \mod i)$-th truth assignment to the literals in the clauses assigned to p_i and the $(k \mod j)$-th truth assignment to the literals in the clauses assigned to p_j are inconsistent within or with each other.

Now note that the above NFA rejects a unary string of length ℓ iff the following are satisfied:

(A) for each $i < s$, $(\ell \mod p_i)$-th truth assignment to the literals in clauses assigned to p_i satisfy the clauses assigned to p_i.

(B) for each $i, j < s$, if clauses assigned to p_i and p_j have a common variable, then $(\ell \mod p_i)$-th and $(\ell \mod p_j)$-th truth assignment to the literals in clauses assigned to p_i and p_j respectively are consistent.

Thus, the language accepted by the above NFA is universal iff the 3SAT formula is not satisfiable. The number of states in the above NFA is bounded by $(3m/r) \cdot c'm$ (for cycles in (1)), plus $(c'm)^2 \cdot 3m$ (for cycles in (2), as there are m variables each appearing at most thrice, so one needs to check at most 3m pairs). So the number of states is proportional to m^3. It follows from the Exponential Time Hypothesis that the complexity of testing universality for n-state NFA is at least $2^{\Omega'(n^{1/3})}$. ◀

▶ Remark 7. One can improve Corollary 6 to computations in **NLOGSPACE** and, by Savitch's Theorem [19], in **DSPACE**$((\log n)^2)$. The reason is that **NLOGSPACE** is closed under complementation, allows to store constantly many states (given by their address in the input) and allows to check whether there is a path from one state to another in a given number of steps (as long as that is bounded by a polynomial). For that reason, one can also check whether, for a number $m \leq n$, one can reach an accepting state in exactly m steps from the start state. This allows to produce a stem of length n from the input where acceptance / rejection is labelled correctly for all of these states; let q be the last state of this stem. Then for each number $m = n+1, \ldots, 2n$, one does the following:

- Check whether there exist numbers h, k such that
  1. $h < k$ and $n+1 \leq m \leq n+k-h$ and m modulo $k-h = k$ modulo $k-h$;
  2. There is an accepting state r reachable from start state in exactly h, k, m steps – note that there is exactly one such accepting run for each of these numbers due to the UFA property;
  3. The run to the state r of length h has no repetition of states;
  4. The run to the state r of length k has some state q' repeated twice and all states visited twice and all states visited before reaching q' or from the second visit of q' onwards are also visited by the run accepting the word of length h;
  5. No state of the run to state r of length k is visited three or more times;
  6. The accepting runs for the words of length k and length m visit the same states.
- If so, then retain h, k, m and continue else go to the next m.
- In the case that h, k, m are found and retained, output a cycle connected by one edge from state q of length $k-h$ which has exactly one accepting state which is visited on reading a word of length m.

This so output automaton is in Chrobak Normal Form and it is also unambiguous due to the input automaton being unambiguous; furthermore, its size is at most quadratic in the size of the input automaton and thus, the complexity class of this algorithm combined with that of Theorem 5 is **NLOGSPACE**.

▶ **Theorem 13.** *Under the assumption of the Exponential Time Hypothesis, one needs at least* $2^{\Omega'(n^{1/4})}$ *time to compute an UFA for the language of the concatenation of the languages of two given* n*-state UFAs in worst case.*

Proof. Recall the proof of Theorem 12 and the values of m, m' there. The cycles of H_1 and H_2 in the proof of Theorem 12 have length $2(m'+1) \cdot p$ for some number p of size $O(m)$ which is either a prime or the constant 1. Furthermore for each number $\ell = 0, 1, \ldots, 2m'+1$, there is at most one cycle A_ℓ in the UFA for H_1 and at most one cycle B_ℓ in the UFA for H_2 which accepts a string of length ℓ modulo $2(m'+1)$ (if any).

Thus one can construct a new intersection automaton of H_1 and H_2 such that it consists of $2(m'+1)$ cycles E_ℓ, where E_ℓ has the length $\mathrm{lcm}(|A_\ell|, |B_\ell|)$ and the t-th state of E_ℓ is accepting iff t has the remainder ℓ when divided by $2(m'+1)$ and both cycles A_ℓ and B_ℓ, after t steps from the start state of those cycles, are in an accepting state in the corresponding automata H_1 and H_2 – here $\mathrm{lcm}(a, b)$ denotes least common multiple of a and b. Thus the so constructed product automaton consists of all cycles E_ℓ and accepts a word iff both H_1 and H_2 accept this word. Furthermore, the automaton, for each $\ell = 0, 1, \ldots, 2m'+1$, for any h, accepts a string of length $\ell + 2(m'+1)h$, only in the cycle E_ℓ, if at all. Thus the constructed automaton is an UFA.

As the size of each cycle E_ℓ is $O(m^3)$ and as there are $2(m'+1)$ cycles and $m' \in \Theta(m)$, the overall size of this automaton is $O(m^4)$. Thus, when n is the size of the product automaton, as the new automaton recognizes $H_1 \cap H_2$, then using Theorem 12, the universality problem for the concatenation of two n-state UFAs is in $2^{\Omega'(n^{1/4})}$ under the assumption of the Exponential Time Hypothesis. ◀

Bounded Simultaneous Messages

Andrej Bogdanov
School of EECS, University of Ottawa, Canada

Krishnamoorthy Dinesh
Dept. of Computer Science and Engineering, Indian Institute of Technology, Palakkad, India

Yuval Filmus
The Henry and Marylin Taub Faculty of Computer Science, Technion, Haifa, Israel

Yuval Ishai
The Henry and Marylin Taub Faculty of Computer Science, Technion, Haifa, Israel

Avi Kaplan
The Henry and Marylin Taub Faculty of Computer Science, Technion, Haifa, Israel

Sruthi Sekar
University of California, Berkeley, CA, USA

Abstract

We consider the following question of *bounded simultaneous messages* (BSM) protocols: Can computationally unbounded Alice and Bob evaluate a function $f(x, y)$ of their inputs by sending polynomial-size messages to a computationally bounded Carol? The special case where f is the mod-2 inner-product function and Carol is bounded to AC^0 has been studied in previous works. The general question can be broadly motivated by applications in which distributed computation is more costly than local computation.

In this work, we initiate a more systematic study of the BSM model, with different functions f and computational bounds on Carol. In particular, we give evidence against the existence of BSM protocols with polynomial-size Carol for naturally distributed variants of NP-complete languages.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Probabilistic computation

Keywords and phrases Simultaneous Messages, Instance Hiding, Algebraic degree, Preprocessing, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.23

Related Version *Full Version*: `https://arxiv.org/abs/2310.00334` [8]

Funding *Andrej Bogdanov*: Supported by an NSERC Discovery Grant.
Yuval Filmus: Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC.
Yuval Ishai: Supported by ERC Project NTSC (742754), ISF grant 2774/20, and BSF grant 2018393.
Avi Kaplan: Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC, and ERC Project NTSC (742754).
Sruthi Sekar: Supported by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc.

Acknowledgements The authors would like to thanks the anonymous reviewers for their comments.

1 Introduction

The simultaneous messages model is a model for evaluating a function $f(x, y)$ of two inputs: Alice, who holds x and Bob, who holds y, simultaneously send messages to a referee Carol, who outputs the value $f(x, y)$. This model was introduced by Yao [42] and further studied

© Andrej Bogdanov, Krishnamoorthy Dinesh, Yuval Filmus, Yuval Ishai, Avi Kaplan, and Sruthi Sekar; licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 23; pp. 23:1–23:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

by Babai et al. [4] in the context of communication complexity, where the interest is to determine the minimum length of Alice's and Bob's messages required to specify f.[1]

We are primarily interested in Carol's *computational* complexity, assuming Alice and Bob are unbounded. Specifically, our objective is to understand which functions require large *bounded* simultaneous messages (BSM) protocols. As the BSM complexity of any function can only be smaller than its standalone computational complexity, explicit constructions of provably hard functions for this model are subject to the usual barriers [35, 1]. Nevertheless, owing to its connections to other complexity notions [33, 24, 28],[2] explicit almost-cubic lower bounds have been obtained for depth-3 AND-OR formulas [27], and almost-quadratic lower bounds for arbitrary AND-OR formulas [40], both witnessed by the "inner product modulo two" (IP) function. It remains a challenge to prove BSM lower bounds for other models in which circuit lower bounds are known such as depth-3 AND-OR circuits [32] and parities of DNF [12].

Motivated by cryptographic applications, Rothblum [36] conjectures that IP has exponential-size bounded-depth unbounded-fan-in BSM complexity. In particular, IP does not admit a BSM protocol with Carol in AC^0. Filmus et al. [16] confirmed the conjecture assuming Alice's message is not much longer than her input.

BSM and secure two-party computation

A primary motivation for revisiting the BSM model stems from its relevance to cryptographically secure two-party computation. In this model, the BSM model is the "ideal model" in which evaluation is carried out by a trusted intermediary Carol. A crowning achievement of modern cryptography is the development of secure two-party protocols that dispense off the need for a trusted Carol [43, 19, 7, 10]. However, the complexity of the ideal model is of interest for the following reasons.

First, despite tremendous progress in the design of two-party protocols for secure function evaluation, the overhead remains substantial. Moreover, certain features of trusted evaluation, such as fairness, are inherently lost by any two-party implementation. In practice, a trusted Carol is often preferred over a two-party protocol, and Carol's computational cost needs to be compensated. BSM complexity captures Carol's minimal computational cost incurred under the best-possible preprocessing of Alice's and Bob's inputs.

Second, in the design of two-party protocols, it can be desirable to preprocess the inputs so as to minimize the complexity of the interactive phase. The complexity measure to be optimized depends on the type of protocol. For instance, in protocols based on oblivious transfer [19, 25, 23], the communication complexity and round complexity of the protocol are proportional to the circuit size and circuit depth of f, respectively. In protocols that rely on fully homomorphic encryption [18, 9] and avoid an expensive "bootstrapping" technique, the computational cost is governed by the degree of f as a polynomial over some underlying finite field. The degree also determines the fraction of corrupted servers that can be tolerated in non-interactive secure computation protocols that employ a set of untrusted servers. In this context, BSM complexity captures the best-possible savings that can be obtained by endowing Alice and Bob with unbounded computational power in the preprocessing phase. Conversely, a lower bound in the BSM model indicates limits of preprocessing in a two-party protocol design.

---

[1] Babai et al. and others [4, 30, 34] more generally study a multi-party variant of this model.

[2] In this literature f is the adjacency function of a bipartite graph and BSM complexity is called *bipartite graph complexity*.

With these motivations in mind, we study the BSM complexity of natural classes of functions that may arise in two-party computation, including Boolean circuits, arithmetic circuits, low-degree polynomials, and time-unbounded halting Turing machines. As some of these classes do not admit explicit lower bounds, any evidence of their hardness must be conditional. One of our contributions is the development of hardness reductions and completeness results for BSM complexity.

Generic constructions and non-explicit bounds

Our primary focus is on Carol's computational complexity, with Alice's and Bob's message size (i.e., communication complexity) as a secondary parameter. Clearly every function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ has a regular circuit size of at most 2^{2n} and therefore BSM circuit size at most 2^{2n} with message size n. We are interested in the regime where Alice's and Bob's messages have a length of at least n.

It is also known that every function has a BSM circuit (parity of ANDs) of size $O(2^n)$ with message size 2^n. This is close to optimal as most functions require BSM circuits of size $\Omega(2^n)$. In contrast, most functions' circuit size is $\Omega(2^{2n}/n)$. Although BSM complexity can be much smaller than circuit complexity, every BSM of circuit size s and total message length ℓ can be simulated by a circuit of size $\ell \cdot 2^n + s$.

1.1 Our Results

Conditional impossibility of efficient BSM for NP

As our first result, we study a distributed variant of SAT, which we refer to as *split SAT*:

$$\mathrm{SSAT} := \{(\alpha, \beta) : \alpha, \beta \text{ are CNFs and } \alpha \wedge \beta \text{ is satisfiable}\}.$$

In Section 2.1 we show that if SSAT has BSM circuit complexity $\mathsf{poly}(|\alpha| + |\beta|)$ then every language in NP can be decided by circuit families of size $O(\mathsf{poly}(n) \cdot 2^{n/2})$. In contrapositive form, no efficient BSM for SSAT exists unless NP has circuits of size $O(\mathsf{poly}(n) \cdot 2^{n/2})$.

We conjecture that the following two NP-languages meet this stringent lower bound:

1. Turing Machine Acceptance: The language ATM_c consists of all pairs (M, x) where M is a nondeterministic Turing Machine that accepts x in $|x|^c$ steps. Here c is a fixed constant; the conjecture is plausible for any $c > 1$. This language is complete in the sense that if $\mathrm{NTIME}(n^c)$ requires circuit size $s(n)$ then ATM_c requires circuit size $s(n - O(1))$.
2. Succinct Subset Sum: The input is an element x of the finite field $\mathbb{F}_{3^m}$. The yes-instances are those x for which there exist coefficients $a_0, a_1, \cdots, a_n \in \{0,1\} \subseteq \mathbb{F}_{3^m}$ such that $a_0 + a_1 x + \cdots + a_n x^n = 0$ and not all a_i are 0, where $n = \lfloor m \log_2 3 \rfloor$.
   While the brute-force 2^n time complexity of worst-case subset-sum can be improved to $2^{n/2}$ in the RAM-model [22, 37, 29], it is unclear if these algorithms yield any savings in circuit size. In contrast, one advantage of the circuit model is that it can decide any language L in size $O(\min(|\{0,1\}^n \cap L|, |\{0,1\}^n \setminus L|))$ by memorizing all yes or no instances of a given length (whichever is smaller). We conjecture that the fraction of yes and no Succinct Subset Sum instances is at least $2^{-n/100}$ thereby rendering this attack ineffective.

As a consequence, in Corollary 5 we show BSM hardness for NP follows from the existence of very strong worst-case one-way functions. The required inversion complexity is as large as $\mathsf{poly}(n) \cdot 2^{n/2}$ under a plausible hardness assumption and $\mathsf{poly}(n) \cdot 2^{2n/3}$ unconditionally (for length-preserving functions). The best-known algorithms for generic function inversion have

conjectured worst-case complexity $\mathsf{poly}(n) \cdot 2^{2n/3}$ in the RAM model with preprocessing [21]. The best unconditional upper bound is $\mathsf{poly}(n) \cdot 2^{3n/4}$ [15]. We do not know any non-trivial circuits for generic inversion.

BSM and Instance Hiding

We consider the notion of instance hiding introduced by Beaver and Feigenbaum [5]. An instance hiding (IH) scheme consists of a primary actor, called Henry, with query access to independent oracles. To compute a function on a given input, Henry may query the oracles, but in such a way that each oracle learns nothing about the input, meaning that the query distribution each oracle sees depends only on the input length. Beaver and Feigenbaum allow interactive oracle queries to multiple oracles; we specialize to non-interactive queries and two oracles.

Fortnow and Szegedy [17] asked whether languages in NP can have an instance hiding scheme where Henry is a polynomial-sized circuit. They showed that SAT cannot have $\mathsf{poly}(n)$-sized IH scheme where the two oracles return 1-bit answers under a standard complexity-theoretic assumption. However, a similar statement with oracles returning 2 or more bits is unknown. The hardness assumption about IH which we make here is that languages in NP do not have a $\mathsf{poly}(n)$-sized instance hiding scheme.

In Proposition 10 we show that if BSM for SSAT has circuit complexity $\mathsf{poly}(|\alpha| + |\beta|)$ then every language in NP has a polynomial-size instance hiding scheme. We interpret this as additional evidence against the existence of efficient BSM for NP. The same holds for the split variants of 3-coloring (Proposition 11) and partition (Proposition 12):

$$\mathrm{SCOL} := \{(A, B) : A, B \text{ are graphs on the same vertex set, } A \cup B \text{ is 3-colorable}\},$$

$$\mathrm{SPRT} := \left\{(A, B) : \begin{array}{c} A, B \text{ are sets of non-negative integers,} \\ A \cup B \text{ can be partitioned into two sets of equal sum} \end{array}\right\}.$$

We achieve this via a notion of a reduction that we call *split-hide* reduction (Definition 7). For a language A, we define a language B which is a "split variant" of A, such that if A split-hide reduces to B and B has a BSM protocol with polynomial sized Carol, then A has IH schemes where Henry is polynomial sized (Lemma 8). In addition, if A is NP-complete, then every language in NP has polynomial sized IH schemes (Corollary 9)

Having established this relation between instance hiding and BSM (in Section 2.3), we also obtain a connection between instance hiding and Private Information Retrieval [11] and use this connection (in the form of reduction between the models) to obtain universal lower bounds on IH schemes (Appendix A.1 [8]). This, in turn, gives a two-way reduction chain between BSM and locally decodable codes (Appendix A.2 [8]). Both of these connections could be of independent interest.

BSM and algebraic polynomials

We now move on to the setting where Carol is an algebraic polynomial over some ring. As explained before in the introduction, this setting is motivated by the design of secure two-party protocols that rely on fully homomorphic encryption [18, 9] and avoids expensive bootstrapping (where the computational cost is dependent on the degree of Carol). Every function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ can be represented by a polynomial of degree at most $2n$, and this is also tight. We ask whether this bound can be reduced if we allow preprocessing. To this end, we define $d_f(n, m)$ to be the minimal degree of a polynomial over $\mathbb{F}_2$ such that there exists a BSM protocol for f with preprocessing length of m bits and Carol computing the polynomial, and we define

$$d(n,m) := \max\{d_f(n,m) \colon f \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\}.$$

We prove a tight bound of $\Theta(n/\log n)$ for the case $m = \mathsf{poly}(n)$. The upper bound was obtained by Beaver et al. [6]. We show a lower bound for the equality function (Proposition 15).

Going ahead, we ask if there are any savings (in terms of degree) in computing the equality function if we allow Carol to be a polynomial over $\mathbb{Z}_k$ where k is composite, followed by a Boolean decision predicate $P : \mathbb{Z}_k \to \{0,1\}$. We show that this is indeed the case for $k = 6$ (Proposition B.27 [8]) using known constructions of matching vectors [20]. In particular, we give a constant degree Carol, but with superpolynomial ($\exp(\widetilde{O}(\sqrt{n}))$) message length over $\mathbb{Z}_6$ which should be contrasted with the BSM protocol over $\mathbb{F}_2$ (Proposition 14) of polynomial message length but with near linear degree.

BSM, Matrix multiplication and RE languages

We now turn to the setting where Carol is an arithmetic circuit as well as a Turing machine. We start with the setting of an arithmetic circuit. The problem of interest is the matrix multiplication over reals. A remarkable result of Strassen [39] showed a non-trivial algorithm for matrix multiplication of two $n \times n$ matrices over reals in $O(n^{\log_2 7})$ time. Since then, there has been a flurry of improvements (notably [13, 14, 2]) where the best value of the exponent in the runtime is denoted by ω. These results also imply an arithmetic circuit performing $O(n^\omega)$ additions and multiplications over reals computing the matrix product. A natural question is: can there be a BSM protocol where Carol is an arithmetic circuit of size $o(n^\omega)$ such that the preprocessing can have a non-trivial saving in size of Carol? We answer this question in negative in Theorem 13.

Finally, we turn to the setting where Carol is a Turing machine. In Proposition 17, we show that any recursively enumerable $f \in \mathsf{RE}$ (viewed as a language) has a BSM protocol where Carol computes a decidable language (in R).

Summary of results

The results are summarized in Table 1 according to Carol's computational power.

- $\mathsf{BSM}(m,s)$ is the class of all functions $f \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that have a two-party SM protocol of size s (namely, Carol is implemented by a circuit of size s) and preprocessing of length m (namely, Alice and Bob send messages of length m; we place a "$\cdot$" symbol in cases where the result is independent of the corresponding parameter).
- $\mathsf{IH}(a,s)$ is the class of all functions $f \colon \{0,1\}^n \to \{0,1\}$ that have a two-oracle Instance Hiding scheme in which Henry's circuit size is s and each oracle answer is of length a.

We use RE and R to denote the recursively enumerable and recursive languages respectively.

1.2 BSM protocols implicit in literature

In Appendix B.1 [8] we discuss BSM protocols for a subclass of Boolean functions that can be obtained readily from known results in the literature. In particular, we consider the bitwise-combined functions which are functions of the form $f(x,y) = g(x * y)$ where g is a Boolean function on n bits and $* \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a *combiner function* that takes in two n bit strings and output an n bit string. Babai *et.al* [4] considers SM protocols, where $*$ is the bitwise XOR combiner function, with a preprocessing length of $O(2^{0.92n})$. Ambainis and Lokam [3] improved this to obtain SM protocols for bitwise XOR combined

Table 1 Summary of Results.

Bound	Result	Reference
Size	$\text{SSAT} \in \mathsf{BSM}(\cdot, \mathsf{poly}(n)) \implies \mathsf{NP} \subseteq \mathsf{SIZE}(\mathsf{poly}(n) \cdot 2^{n/2})$	Thm. 1
	$\implies$ No $\widetilde{O}(2^{2n/3})$-secure length preserving OWF	Cor. 5
	$\text{SSAT/SCOL/SPRT} \in \mathsf{BSM}(\cdot, \mathsf{poly}(n)) \implies \mathsf{NP} \subseteq \mathsf{IH}(\cdot, \mathsf{poly}(n))$	Thm. 6
	$g(x \oplus y) \in \mathsf{BSM}(\cdot, s) \implies g \in \mathsf{SIZE}((s/n) \cdot 2^{n/2})$	Prop. B.2 [8]
	$g(x \vee y), g(x \wedge y) \in \mathsf{BSM}(2^{0.729n}, \cdot)$ for all functions $g\colon \{0,1\}^n \to \{0,1\}$	Thm. B.25 [8]
Arithmetic	Matrix multiplication with preprocessing requires $\Omega(n^\omega)$ operations	Thm. 13
Degree	$d(n,m) = O(n/\log(m/n))$ for $n < m$	Prop. 14
	$d_{\text{EQ}}(n,m) = \Omega(n/\log m)$ for $n \leq m$	Prop. 15
	$d(n, n^{C+1}) = \Theta(n/\log n)$ ($C > 0$ constant)	Cor. 16
	$d_{\text{EQ}}(n,m) = 2$ for $m = 2^{\widetilde{\Theta}(\sqrt{n})}$ (over $\mathbb{Z}_6$)	Prop. B.27 [8]
Recursive	$\mathsf{RE} \subseteq \mathsf{BSM}(2n+1, \mathsf{R})$	Prop. 17

functions where the preprocessing length is bounded by $O(2^{0.729n})$. A standard counting argument shows that most XOR combined functions require a preprocessing length of $\Omega(2^{0.5n})$. (Proposition B.8 [8]). The XOR combined functions are interesting since an efficient BSM protocol computing f would give an efficient IH scheme for g (Proposition B.1 [8]). Using this connection (in Appendix B.1.2 [8]), we could rule out bitwise-XOR combined version of NP-hard languages from having $\mathsf{poly}(n)$ sized BSM protocols with message complexity of $n+1$ unless the polynomial hierarchy collapse and bitwise-XOR combined version PSPACE-hard languages having $\mathsf{poly}(n)$ sized BSM protocols with a message complexity of $n + O(\log n)$ unless $\mathsf{PSPACE} \subseteq \mathsf{PP}^{\mathsf{NP}}/\mathsf{poly}$. These conclusions rely on structural complexity results about instance hiding due to Fortnow and Szegedy [17] and Tripathi [41].

We continue the study where the bitwise combiner function is a bitwise OR as well as bitwise AND function and obtain SM protocols with a preprocessing length of $O(2^{0.729n})$ matching the result of Ambainis and Lokam (Theorem B.25 [8]) for XOR combined functions [3].

1.3 Paper organization

The overview for the rest of the paper is as follows. We study the BSM model under computational restrictions on Carol. In Section 2, we restrict the Carol to be a polynomial-sized circuit, and show that the Split-SAT problem (SSAT) is unlikely to have polynomial sized BSM protocols and we extend the same to two other distributed variants of NP-complete problems. In Section 3, we restrict Carol to be an arithmetic circuit and in Section 4, we restrict Carol to be an algebraic polynomial over $\mathbb{F}_2$. In Section 5, we discuss the setting where Carol is a Turing machine and show that it is possible to decide any RE language (including the Halting problem) with preprocessing.

The full version [8] of the paper includes appendices omitted in this version due to space constraints. In Appendix A [8], we provide connections of BSM to related models. In particular, we exploit this connection to argue that most Boolean functions do not have efficient instance hiding schemes. In Appendix B [8], we discuss a few BSM protocols that are implicit in the existing literature.

2 Bounded Simultaneous Messages for NP problems

In this section we show that the existence of BSM for split variants of certain NP complete problems yields unlikely consequences, including better than brute-force circuits for all NP languages (Section 2.1), one-way functions (Section 2.2), and instance hiding schemes for all of NP (Section 2.3).

2.1 Split SAT and NP languages

▶ **Theorem 1.** *If* SSAT *has a 2-party BSM protocol of size $s(n)$ and message length $\ell(n)$ then every L in* NP *has circuit size $O(\ell(p(n)) \cdot 2^{n/2}) + s(p(n))$ for some polynomial p.*

Proof. We will assume without loss of generality that L rejects all inputs of odd length. This can be enforced using the encoding

$$L = \{0x \colon x \text{ is a yes instance of odd length}\} \cup \{11x \colon x \text{ is a yes instance of even length}\}.$$

Consider the following algorithm for L:

Preprocessing. Given an even input length n:

1. Construct the Cook-Levin CNF ϕ which accepts (x, y, z) for some z if and only if $xy \in L$ with $|x| = |y| = n/2$.
2. For every x with $|x| = n/2$, construct the CNF

   $$\alpha_x(u, v, z) = \text{“}(x = u) \wedge \phi(x, v, z)\text{”}, \qquad \text{where } |u| = |v| = n/2,$$

   then determine and store the message a_x that Alice sends in the BSM SSAT protocol when her input is α_x.
3. For every y with $|y| = n/2$, construct the CNF

   $$\beta_y(u, v, z) = \text{“}(y = v) \wedge \phi(u, y, z)\text{”}, \qquad \text{where } |u| = |v| = n/2,$$

   then determine and store the message b_y that Bob sends in the BSM SSAT protocol when his input is β_y.

Execution. On input xy with $|x| = |y| = n/2$, simulate Carol on messages a_x and b_y and output her answer.

Under the assumptions of the theorem this algorithm decides L because $\alpha_x \wedge \beta_y$ is satisfiable if and only if $\phi(x, y, \cdot)$ is. If $\phi(x, y, z)$ accepts then so does $\alpha_x(x, y, z) \wedge \beta_y(x, y, z)$, while if $\alpha_x(u, v, z) \wedge \beta_y(u, v, z)$ accepts it must be that $x = u$ and $y = v$ so $\phi(x, y, z)$ must also accept.

The circuit representation of this algorithm consists of two tables of size $\ell(p(n)) \cdot 2^{n/2}$, a proportional amount of hardware to select the messages a_x and b_y, and a copy of Carol's circuit, giving the bound on circuit size. ◀

▶ **Corollary 2.** *If there exists a language in* NP *that is not computable by any circuit family of size $\widetilde{O}(2^{n/2})$, then* SSAT *doesn't have a 2-party,* poly(n)*-size BSM protocol.*

2.2 BSM hardness from worst-case one-way functions

We derive consequences of Corollary 2 to the worst-case insecurity of one-way functions.

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is length-preserving if $|f(x)| = |x|$ for all sufficiently long x. It is non-poly-shrinking if $|f(x)| \geq |x|^\epsilon$ for all sufficiently long x and some constant ϵ.

▶ **Proposition 3.** *If every* NP *problem has circuit size* $\mathsf{poly}(n) \cdot 2^{n/2}$ *then every efficient length-preserving function can be inverted on every output in size* $\mathsf{poly}(n) \cdot 2^{2n/3}$.

Proof. Let L be the language

$$L = \{(y, a)\colon \text{there exists } x \text{ such that } f(x) = y \text{ and } a \text{ is a prefix of } x\}.$$

By the assumption L has circuits C of size $\mathsf{poly}(|y| + |a|) \cdot 2^{(|y|+|a|)/2}$, which is at most $\mathsf{poly}(n) \cdot 2^{2n/3}$ assuming $|y| = n$ and $|a| \leq n/3$.

Given y in the image of f, the circuit C can be applied iteratively to find the first $n/3$ bits of a preimage x. Once those are found, the other $2n/3$ bits can be computed by brute-force search. The resulting preimage finder has circuit size $\mathsf{poly}(n) \cdot 2^{2n/3}$. ◀

▶ **Proposition 4.** *If every* NP *problem has circuit size* $\mathsf{poly}(n) \cdot 2^{n/2}$ *but* $\mathsf{E}_{\|}^{\mathsf{NP}}$ *requires exponential size nonadaptive SAT-oracle circuits then every efficient non-poly-shrinking function can be inverted on every output of length* n *in size* $\mathsf{poly}(n) \cdot 2^{n/2}$.

Proof. Klivans and van Melkebeek [26] (see also the discussion in [38]) show that under the second assumption there is an efficient deterministic algorithm that on input y of length n produces a list of circuits $V_1, \cdots, V_{\mathsf{poly}(n)}$ such that at least one of the circuits V_i accepts a *unique* x for which $f(x) = y$, assuming y is in the image of the function f. Let

$$L = \{(y, i, j)\colon \text{there exists } x \text{ for which } V_i(x) \text{ accepts and } x_j = 1\}.$$

As L is an NP language, by the first assumption, it has circuits C of size $\mathsf{poly}(|y| + |i| + |j|) \cdot 2^{(|y|+|i|+|j|)/2} = \mathsf{poly}(n) \cdot 2^{n/2}$. Using C, a preimage of y can be found among the rows of the table $x_{ij} = C(y, i, j)$. ◀

▶ **Corollary 5.** *If SSAT has a 2-party,* $\mathsf{poly}(n)$*-size BSM protocol then every efficiently computable function can be inverted on length-*n *outputs in (1) size* $\mathsf{poly}(n) \cdot 2^{2n/3}$ *assuming it is length-preserving; (2) size* $\mathsf{poly}(n) \cdot 2^{n/2}$ *assuming it is non-poly-shrinking and* $\mathsf{E}_{\|}^{\mathsf{NP}}$ *requires exponential size nonadaptive SAT-oracle circuits.*

2.3 Instance hiding and NP languages

A two-query *instance hiding (IH) scheme* [5] consists of size circuit (Henry) that receives an input and its output using queries to two oracles (Alice and Bob) such that the distribution of each query depends only on Henry's input length.

The instance hiding scheme is efficient if the size of Henry is polynomial in its input length. Our main result in this section is the following:

▶ **Theorem 6.** *If there is an efficient BSM protocol to one of the languages* SSAT*,* SCOL*,* SPRT*, then every language in* NP *has an efficient IH scheme.*

The connection between IH an BSM will be established using the following notion of reduction.

▶ **Definition 7.** *Let L_1, L_2 be a languages. A* split-hide reduction from L_1 to L_2 *is a pair of polynomial-time computable randomized mappings $a, b\colon \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(n)}$ satisfying*
Correctness: $x \in L_1 \iff a(x)b(x) \in L_2$.
Privacy: The marginal distributions of $a(x)$ and $b(x)$ depend only on the length of x.
If such a reduction exists, we say that L_1 split-hide reduces to L_2*, and denote $L_1 \leq_p^{\mathsf{sh}} L_2$.*

▶ **Lemma 8.** *Let L_1, L_2 be two languages. If $L_1 \leq_p^{\mathsf{sh}} L_2$ and L_2 has an efficient BSM protocol, then L_1 has an efficient IH scheme with a single nonadaptive pair of queries.*

Proof. Suppose that L_2 has a 2-party efficient BSM protocol implemented by Alice, Bob, and Carol, and that $L_1 \leq_p^{\mathsf{sh}} L_2$ using mappings a and b. In the IH for L_1, on input x, Henry submits queries $a(x)$ and $b(x)$ to Alice and Bob, respectively, forwards their answers to Carol, and produces her output. ◀

▶ **Corollary 9.** *Let L_1, L_2 be a languages, and suppose that L_1 is NP-hard. If $L_1 \leq_p^{\mathsf{sh}} L_2$ and L_2 has an efficient BSM protocol, then every language in NP has an efficient IH scheme with a single nonadaptive pair of queries.*

Here, NP-hardness is assumed to hold under reductions that map all instances of a given length n into instances of the same length $m(n)$.

Proof. In the IH scheme for L_1, Henry implements the reduction from L_1 to L_2 and then runs the IH for L_2 from Lemma 8. ◀

Theorem 6 follows from Corollary 9 and the propositions below which show that SAT, 3COL, and PARTITION each split-hide reduce to their split variant.

▶ **Proposition 10.** SAT $\leq_p^{\mathsf{sh}}$ SSAT.

Proof. By the standard reduction from SAT to 3SAT we can represent the input instance by a 3CNF $\varphi(x_1, \ldots, x_n)$, where n is chosen large enough to embed all SAT instances of a given size. We describe the randomized mappings a and b that yield CNFs α and β, respectively, in variables $x_1, \ldots, x_n$ and additional auxiliary variables $y_1, \ldots, y_{2N}$, where $N = 8\binom{n}{3}$. Let $C_1, \ldots, C_N$ be a list of all possible clauses of width at most 3 over the n variables $x_1, \ldots, x_n$. Let π be a random permutation of $[2N]$. We define

$$\alpha = \bigwedge_{i \in [N]} (C_i \vee y_{\pi(i)}).$$

Let $I = \{i : C_i \in \varphi\}$ and $J = \{N + i : C_i \notin \varphi\}$. We define

$$\beta = \bigwedge_{i \in I \cup J} \neg y_{\pi(i)}.$$

Correctness: We argue that φ is satisfiable if and only if $\alpha \wedge \beta$ is satisfiable.

- Suppose that φ is satisfiable. Then, we can satisfy α by taking the satisfying assignment of φ to satisfy each clause $(C_i \vee y_{\pi(i)})$ for $i \in I$, and assign a true value to $y_{\pi(i)}$ for every $i \in [N] \setminus I$. To satisfy β, assign a false value to $y_{\pi(i)}$ for every $i \in I$, which will not affect α's satisfiability, and assign a false value to $y_{\pi(i)}$ for every $i \in J$.
- Suppose that $\alpha \wedge \beta$ is satisfiable. Since β is satisfiable, it follows that the satisfying assignment assigns false to $y_{\pi(i)}$ for every $i \in I$, which implies that α can be satisfied by the same assignment only if C_i is satisfied for every $i \in I$, namely φ itself is satisfied.

Privacy: The y-variables in both α and β are random length N subsequences of $(y_1, \ldots, y_{2N})$. ◀

▶ **Proposition 11.** 3COL $\leq_p^{\mathsf{sh}}$ SCOL.

Proof. Given a graph $G = (V, E)$ with vertices $V = \{1, \ldots, n\}$ the reduction produces the following two graphs A and B on the same vertex set $\tilde{V}$. Let $N = \binom{n}{2}$ and π be a random permutation of $[2N]$. We identify $[N]$ with ordered pairs of vertices in V.

- Vertices $\tilde{V}$: For each $v \in V$ there is a vertex v in $\tilde{V}$, and for each $j \in [2N]$ there are vertices $a_j, b_j, c_j, a'_j, b'_j, c'_j$.
- Edges in A: For each pair $(v < v') \in V$, let $j = \pi(v, v')$. We connect v to a_j, b_j, v' to a'_j, b'_j, and add triangles (a_j, b_j, c_j) and (a'_j, b'_j, c'_j).
- The edges in B are $\{(c_{\pi(e)}, c'_{\pi(e)}) : e \in E\} \cup \{(c_{\pi(N+e)}, c'_{\pi(N+e)}) : e \notin E\}$.

Correctness: If G is 3-colorable, the 3-coloring can be extended to a valid coloring of $A \cup B$ by coloring c_j and c'_j for $j = \pi(v, v')$ with the colors of v and v', respectively, and coloring a_j, b_j, a'_j, b'_j with the remaining colors. The vertices indexed by j that are not in the image of $\pi([N])$ are assigned fixed consistent colors, e.g., $(a_j, b_j, c_j, a'_j, b'_j, c'_j) \to (\mathtt{R}, \mathtt{B}, \mathtt{G}, \mathtt{B}, \mathtt{G}, \mathtt{R})$. Conversely, if $A \cup B$ is 3-colorable, v and c_j must have the same color and so must v' and $c_{j'}$; by the definition of B the coloring on V is valid for G.

Privacy: The graph A consists of N pairs of "diamonds" v, a_j, b_j, c_j and v', a'_j, b'_j, c'_j, one for each vertex-pair $v < v'$, where j is randomly assigned one of N distinct values in $[2N]$. The graph B is a matching chosen at random among those that match N out of the $2N$ pairs $(c_1, c'_1), \ldots, (c_{2N}, c'_{2N})$. Both distributions depend on G only through n. ◀

▶ **Proposition 12.** PARTITION $\leq_p^{\mathsf{sh}}$ SPRT.

Proof. The input to PARTITION problem consists of a multi-set $S = \{x_1, \ldots, x_n\}$ of n non-negative integers (not necessarily distinct) of n-bits forming a multi-set S.

Reduction: Set $p = n2^n + 1$. For each $i \in \{0, 1, \ldots, n-1\}$, pick a y_i uniformly at random from $\{0, 1, \ldots, p-1\}$ and let z_i be the unique residue $(x_i - y_i) \mod p$. Define

$$\begin{aligned} Y_i &= y_i \cdot 8^n + 1 \cdot 8^i \\ Z_i &= z_i \cdot 8^n + 2 \cdot 8^i \\ W_i &= 3 \cdot 8^i \end{aligned}$$

Alice's and Bob's inputs are the sets

$$\begin{aligned} A &= \{Y_0, \ldots, Y_{n-1}\} \\ B &= \{Z_0, \ldots, Z_{n-1}\} \cup \{W_0, \ldots, W_{n-1}\} \cup E, \qquad E = \{2^j \cdot p8^n \mid 0 \leq j \leq \log n\}. \end{aligned}$$

Correctness: We show that $U = A \cup B$ has a partition $\sum C = \sum D$ if and only if S has a partition $\sum P = \sum Q$, where $\sum X$ is the sum of all elements in a multi-set X. Let $U' = \{Y_i, Z_i, W_i \mid 0 \leq i \leq n-1\}$ so that $U' \cup E = U = A \cup B$. We show the following three statements are equivalent. Correctness follows from the equivalence of 1 and 3.

1. S has a partition $\sum P = \sum Q$
2. U' has a partition $\sum C' \equiv \sum D' \bmod p8^n$
3. U has a partition $\sum C = \sum D$.

$1 \to 2$: Assuming $\sum P = \sum Q$, For each $x_i \in P$, place Y_i, Z_i in C' and W_i in D'. For each $x_i \in Q$, place W_i in C' and Y_i, Z_i in D'. This ensures that $\sum C' - \sum D'$ is a multiple of 8^n and moreover $\sum C' - \sum D' = (\sum P - \sum Q)/8^n$. With P and Q of equal sums, $\sum C' - \sum D' \equiv 0 \mod p8^n$, we can conclude that C', D' is the desired partition of U'.

$2 \to 1$: Assume $\sum C' \equiv \sum D' \bmod p8^n$. Then for every i (1) $Y_i, Z_i \in C'$ and $W_i \in D'$, or (2) $Y_i, Z_i \in D'$ and $W_i \in C'$, or (3) Y_i, Z_i, W_i do not appear in C' and D'. In all other cases the i-th least significant entry in the base-8 representation of $\sum C'$ and $\sum D'$ cannot match.

Assign every number y_i, z_i, w_i to P or Q depending on whether Y_i, Z_i, W_i is in C' or D'. Then $\sum C' \equiv 8^n \sum P + \sum 3 \cdot 8^i$ and $\sum D' \equiv 8^n \sum Q + \sum 3 \cdot 8^i$ modulo $p8^n$, where the second summation is over those indices i that satisfy (1) or (2). Therefore $\sum P \equiv \sum Q$ modulo p. By our choice of p, $\sum P = \sum Q$.

$2 \to 3$: If $\sum C' \equiv \sum D' \bmod p8^n$, then $\sum C' - \sum D' = kp8^n$ for some $-n \le k \le n$. Let E_k be the (unique) subset of E that sums to $|k|\, p8^n$. If $k > 0$ set $C = C'$, $D = D' \cup E_k$. Otherwise, set $C = C' \cup E_k$, $D = D'$. In either case $\sum C = \sum D$.

$3 \to 2$: C' and D' are obtained from C and D by dropping the elements in E.

Privacy: The marginal distributions of y_i as well as z_i are independent of the input instance S and depends only on n. The set A consists of Y_i and B consists of Z_i, W_i and some fixed additional elements none of which depend on the set S. Hence, the marginal distributions A and B are also independent of S and depends only on n. ◀

3 Arithmetic circuits for Matrix Multiplication

An *arithmetic BSM protocol* for a polynomial $p(X, Y)$ is a decomposition of the form $p(X, Y) = \mathrm{Carol}(\mathrm{Alice}(X), \mathrm{Bob}(Y))$. The complexity of the protocol is the smallest possible arithmetic circuit complexity of Carol.

We show that BSM protocols do not help for circuit multiplication. Recall that the *tensor rank* $R(n)$ of $n \times n$ matrix multiplication is the smallest possible number of terms in a decomposition of XY as a sum of products of linear functions in X and Y, respectively. Asymptotically, $\log_n R(n)$ converges to the matrix multiplication exponent ω.

▶ **Theorem 13.** *In any arithmetic BSM protocol for multiplying $n \times n$ matrices Carol must use at least $R(n)/2$ multiplication gates.*

As matrix multiplication can be realized in complexity $O(R(n))$ without preprocessing, BSM does not offer any savings in this setting.

Proof. We prove the contrapositive: A BSM protocol in which Carol uses t multiplication gates yields a representation of XY of tensor rank at most $2t$.

For a polynomial P in the entries of both X and Y, let $c(P)$ be the constant part, $a(P)$ be the linear part involving entries from X, $b(P)$ be the linear part involving entries from Y, and $ab(P)$ be the bilinear part, where each monomial is a product of an entry of X and an entry of Y.

We construct a circuit that computes the low-degree part $\ell(C) = (c(C), a(C), b(C), ab(C))$ using at most $2t$ multiplication gates inductively over the size of Carol's circuit C. For the base case, Carol has size zero its output must come either from Alice of from Bob, so ab must be zero and $q(C)$ is a linear function of X or Y requiring no multiplications. For the inductive step, the following rules show that computing $\ell(P + Q)$ from $\ell(P)$ and $\ell(Q)$ takes no extra multiplications, while computing $\ell(PQ)$ takes at most two extra multiplications (underlined):

$c(P+Q) = c(P) + c(Q)$	$c(PQ) = c(P)c(Q)$
$a(P+Q) = a(P) + a(Q)$	$a(PQ) = c(P)a(Q) + a(P)c(Q)$
$b(P+Q) = b(P) + b(Q)$	$b(PQ) = c(P)b(Q) + b(P)c(Q)$
$ab(P+Q) = ab(P) + ab(Q)$	$ab(PQ) = c(P)ab(Q) + ab(P)c(Q) + \underline{a(P) \cdot b(Q)} + \underline{b(P) \cdot a(Q)}$

As Carol's output is some linear combination of its multiplication gates, $\ell(XY)$ can be computed using at most twice the number of multiplications used by Carol as desired. ◀

4 Polynomials of bounded degree

Every Boolean function $f\colon \{0,1\}^N \to \{0,1\}$ can be represented by a multilinear polynomial of degree at most N over the reals. In this section we ask whether we can reduce the degree by means of preprocessing. Given a function $f\colon \{0,1\}^{2n} \to \{0,1\}$, let $d_f(n,m)$ be the least degree of a polynomial Carol for which $f(x,y) = \mathrm{Carol}(\mathrm{Alice}(x), \mathrm{Bob}(y))$ and Alice, Bob output at most m bits each.

This type of representation provides dramatic savings for $\mathrm{AND}(x,y) = x_1 \cdot \dots \cdot x_n \cdot y_1 \cdot \dots \cdot y_n$, which has degree n, yet $d_{\mathrm{AND}}(n,1) \le 2$. This separation is best possible because the only functions with $d_f(n,m) = 1$ are direct sums.

Let $d(n,m) := \max\left\{d_f(n,m) \colon f\colon \{0,1\}^{2n} \to \{0,1\}\right\}$. Beaver at al. [6] showed (in a different context) that for any constant $C > 0$,

$$d(n, n^{C+1}) \le \frac{2n}{C \log n}.$$

We extend their argument to obtain the following bound.

▶ **Proposition 14.** *For $n < m < 2^n$, it holds that $d(n,m) = O(n/\log(m/n))$.*

Proof. Let $f\colon \{0,1\}^{2n} \to \{0,1\}$, and let p be a multilinear polynomial of degree $2n$ that computes f:

$$p(z_1, \dots, z_{2n}) = \sum_{S \subseteq [2n]} c_S \cdot z^S.$$

Consider now the following strategy to reduce the degree of p: split the input z into n/t disjoints set of coordinates, each of size t, for some chosen $t \le n$. For each coordinate subset $T \subseteq [2n]$ of size T, we define $2^t - 1$ variable as follows: for each $\emptyset \ne J \subseteq T$:

$$Z_J = \prod_{i \in J} z_i.$$

It follows that we can express p as a polynomial of degree $\lceil 2n/t \rceil$ by taking each monomial z^S and replacing it with a product of Z_J's for proper choices of J's.

We can now establish a BSM protocol for f with Carol computing a degree-$(2n/t)$ polynomial by letting Alice and Bob compute the Z_J's and send the results to Carol. This requires preprocessing output length of at most $\lceil n/t \rceil \cdot (2^t - 1)$. Choosing

$$t = \lceil \log(m/(4n)) \rceil \le \log(m/(4n)) + 1 = \log(m/n) - 1 \le \log(2^n/n) < n$$

implies preprocessing of length at most

$$\lceil n/t \rceil \cdot (2^t - 1) \le (n/t + 1) \cdot 2^t \underset{t \le n}{\le} (2n/t) \cdot 2^t = (2n/t) \cdot 2^{\lceil \log(m/(4n)) \rceil} \le m/t \underset{t \ge 1}{\le} m.$$

Thus,

$$d(n,m) = \lceil 2n/t \rceil \le 2n/t + 1 = \frac{2n}{\lceil \log(m/(4n)) \rceil} + 1 \le \frac{2n}{\log(m/(4n))} + 1 = O(n/\log(m/n)). \qquad ◀$$

The following proposition about the equality function shows that this bound is almost tight.

▶ **Proposition 15.** *For $n \le m$, $d_{\mathrm{EQ}}(n,m) = \Omega(n/\log m)$.*

Proof. Let us start by considering the case in which the $\mathbb{F}_2$-polynomial has *individual degree* 1, where individual degree-1 means that each variable taken from either Alice's message or Bob's message occurs at most once in any monomial computed by Carol. This means that it is of the form

$$P(z,w) = \sum_{ij} c_{ij} z_i w_j + \sum_i d_i z_i + \sum_j e_j w_j + r.$$

Now consider the n-bit equality function EQ. Suppose that it can be calculated using preprocessing of length m. That is, there exist functions $A, B\colon \{0,1\}^n \to \{0,1\}^m$ and a polynomial P as above such that for all $x, y \in \{0,1\}^n$:

$$\mathrm{EQ}(x,y) = P(A(x), B(y)).$$

For each $x \in \{0,1\}^n$, $Q_x(y) = P(A(x), B(y))$ is an affine form on m variables. Any $m+2$ such forms must be linearly dependent. Hence if $m+2 \le 2^n$, we can find $x_0, \ldots, x_m$ such that

$$Q_{x_0}(y) = Q_{x_1}(y) + \cdots + Q_{x_m}(y).$$

Substituting $y = x_0$, we obtain a contradiction, since the left-hand side should be 1, while the right-hand side is a sum of zeroes. We conclude that $m \ge 2^n - 1$. (This should be compared to the obvious upper bound in which $m = 2^n$.)

Now suppose that $\mathrm{EQ}(x,y) = P(A(x), B(y))$, where P is an arbitrary polynomial of degree d, and $|A(x)| = |B(x)| = m$. Let $A'(x)$ consist of all ANDs of up to d elements from $A(x)$, and define $B'(x)$ similarly. Thus, $|A'(x)| = |B'(x)| = \binom{m}{\le d}$. We can find a polynomial P' of individual degree 1 such that $P'(A'(x), B'(y)) = P(A(x), B(y))$. Therefore,

$$2^n - 1 \le \binom{m}{\le d} < (m+1)^d \implies d \ge \frac{n}{\log(m+1)}. \qquad \blacktriangleleft$$

Combining the last two propositions, we have a tight bound when m is polynomial in n:

▶ **Corollary 16.** *For every constant $C > 1$, $d(n, n^C) = \Theta(n/\log n)$.*

5 Computability with preprocessing

Language $L \subseteq \{0,1\}^* \times \{0,1\}^*$ is BSM-computable with message size $m(n)$ if there is a BSM protocol for L in which

1. Carol is a Turing Machine that receives $m(|x|)$ and $m(|y|)$-bit messages from Alice and Bob on input (x,y)
2. Carol halts on all inputs of the form Alice(x), Bob(y).

In this setting Alice and Bob do not know each other's input length.

▶ **Proposition 17.** *Every recursively enumerable language is BSM-computable with message size $2n+1$.*

As BSM-computability is closed under complement, the proposition extends to co-recursively enumerable languages.

Proof. Let M be a Turing Machine that recognizes L, i.e., it halts on yes-instances only. The following BSM protocol computes L:

- On input x, Alice sends Carol x and the number of pairs (x, y') with $|y'| \le |x|$ such that $xy' \in L$. On input y, Bob sends Carol y and the number of pairs (x', y) with $|x'| \le |y|$ such that $x'y \in L$.

- If $|x| \leq |y|$, Carol runs M on all inputs xy' for $|y'| \leq |x|$ in parallel and halts whenever the number of accepted instances reaches the value sent by Alice. Carol accepts iff (x, y) is among them. If $|x| < |y|$, Alice's and Bob's roles are reversed. ◀

If condition 2 in the definition of BSM-computability were replaced with the stronger requirement that Carol halts on all inputs, Proposition 17 would no longer be true: It is possible to construct a recursively enumerable language that is not BSM-computable in this stronger sense whenever $m(n) = o(2^{n/2})$ by diagonalizing against every possible Carol [31].

6 Conclusion and Open Problems

In this work, we initiate a systematic study of the BSM complexity of several natural classes of functions with different assumptions on Carol: size-bounded, degree-bounded, and arithmetic (summarized in Table 1). Our work suggests several natural open problems.

- *Size-bounded Carol.* Can preprocessing help improve the matrix multiplication time when Carol is a Boolean circuit? Our negative result only considered an arithmetic Carol over the reals (Theorem 13). To argue against the possibility of BSM with polynomial-size Carol for arbitrary NP languages, we suggested two explicit candidates for NP languages with the highest possible circuit complexity: Turing Machine Acceptance and Succinct Subset Sum (see Section 1.1 for the exact conjectures). Studying these conjectures and proposing other natural candidates for maximally hard NP languages may be of independent interest. A final open question is proving an analog of Theorem 1 for SCOL, SPRT or split-versions of some other NP complete languages, where the natural Cook-Levin approach does not seem to apply.
- *Depth-bounded Carol.* We do not have any results pertaining to a depth-bounded Carol. However, the prior result of [16] which shows a negative result for a BSM protocol for the mod-2 inner product function by AC^0 circuits, leaves a non-trivial open question to solve. Can we prove even a weak unconditional lower bound for the mod-3 inner product by AC^0 circuits with mod-2 gates?

References

1. Scott Aaronson and Avi Wigderson. Algebrization: a new barrier in complexity theory. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 731–740, 2008. doi:10.1145/1374376.1374481.
2. 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 2021, Virtual Conference, January 10–13, 2021*, pages 522–539, 2021. doi:10.1137/1.9781611976465.32.
3. Andris Ambainis and Satyanarayana V. Lokam. Improved upper bounds on the simultaneous messages complexity of the generalized addressing function. In *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *Lecture Notes in Computer Science*, pages 207–216, 2000. doi:10.1007/10719839_21.
4. László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication complexity of simultaneous messages. *SIAM J. Comput.*, 33(1):137–166, 2003. doi:10.1137/S0097539700375944.
5. Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *STACS 90, 7th Annual Symposium on Theoretical Aspects of Computer Science, Rouen, France, February 22-24, 1990, Proceedings*, pages 37–48, 1990. doi:10.1007/3-540-52282-4_30.

6 Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Locally random reductions: Improvements and applications. *J. Cryptol.*, 10(1):17–36, 1997. doi:10.1007/s001459900017.

7 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988. doi:10.1145/62212.62213.

8 Andrej Bogdanov, Krishnamoorthy Dinesh, Yuval Filmus, Yuval Ishai, Avi Kaplan, and Sruthi Sekar. Bounded simultaneous messages, 2023. arXiv:2310.00334.

9 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM J. Comput.*, 43(2):831–871, 2014. doi:10.1137/120868669.

10 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19, 1988. doi:10.1145/62212.62214.

11 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50, 1995. doi:10.1109/SFCS.1995.492461.

12 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 47–58, 2016. doi:10.1145/2840728.2840734.

13 Don Coppersmith and Shmuel Winograd. On the asymptotic complexity of matrix multiplication (extended summary). In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 82–90, 1981. doi:10.1109/SFCS.1981.27.

14 A. M. Davie and A. J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, 143(2):351–369, 2013. doi:10.1017/S0308210511001648.

15 Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM Journal on Computing*, 29(3):790–803, 2000. doi:10.1137/S0097539795280512.

16 Yuval Filmus, Yuval Ishai, Avi Kaplan, and Guy Kindler. Limits of preprocessing. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 17:1–17:22, 2020. doi:10.4230/LIPIcs.CCC.2020.17.

17 Lance Fortnow and Mario Szegedy. On the power of two-local random reductions. In *Advances in Cryptology – ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 346–351, 1991. doi:10.1007/3-540-57332-1_29.

18 Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 169–178. ACM, 2009. doi:10.1145/1536414.1536440.

19 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987. doi:10.1145/28395.28420.

20 Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. *Comb.*, 20(1):71–86, 2000. doi:10.1007/s004930070032.

21 M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980. doi:10.1109/TIT.1980.1056220.

22 Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, April 1974. doi:10.1145/321812.321823.

23 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David A. Wagner, editor, *Advances in Cryptology – CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008. doi:10.1007/978-3-540-85174-5_32.

24 Stasys Jukna. On graph complexity. *Comb. Probab. Comput.*, 15(6):855–876, 2006. doi:10.1017/S0963548306007620.

25 Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31. ACM, 1988. doi:10.1145/62212.62215.

26 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

27 Satyanarayana V. Lokam. Graph complexity and slice functions. *Theory Comput. Syst.*, 36(1):71–88, 2003. doi:10.1007/s00224-002-1068-0.

28 Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Found. Trends Theor. Comput. Sci.*, 4(1-2):1–155, 2009. doi:10.1561/0400000011.

29 Jesper Nederlof and Karol Wegrzycki. Improving Schroeppel and Shamir's algorithm for subset sum via orthogonal vectors. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1670–1683, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451024.

30 Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1):211–219, 1993. doi:10.1137/0222016.

31 P. Odifreddi. *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers*. ISSN. Elsevier Science, 1992. URL: https://books.google.co.in/books?id=ahBLqshTFMwC.

32 Ramamohan Paturi, Michael E. Saks, and Francis Zane. Exponential lower bounds for depth three boolean circuits. *Comput. Complex.*, 9(1):1–15, 2000. doi:10.1007/PL00001598.

33 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Informatica*, 25(5):515–535, 1988. doi:10.1007/BF00279952.

34 Pavel Pudlák, Vojtech Rödl, and Jirí Sgall. Boolean circuits, tensor ranks, and communication complexity. *SIAM Journal on Computing*, 26(3):605–633, 1997. doi:10.1137/S0097539794264809.

35 Alexander A. Razborov and Steven Rudich. Natural proofs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 204–213, 1994. doi:10.1145/195058.195134.

36 Guy N. Rothblum. How to compute under AC^0 leakage without secure hardware. In *Advances in Cryptology – CRYPTO 2012 – 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569, 2012. doi:10.1007/978-3-642-32009-5_32.

37 Richard Schroeppel and Adi Shamir. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM Journal on Computing*, 10:456–464, 1981.

38 Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *computational complexity*, 15(4):298–341, 2006.

39 Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, August 1969. doi:10.1007/bf02165411.

40 Avishay Tal. Formula lower bounds via the quantum method. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472.

41 Rahul Tripathi. On the computational power of locally random reductions. *A preliminary version appeared in MFCS 2007*, 2010. URL: https://api.semanticscholar.org/CorpusID:17801736.

42 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 – May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979. doi:10.1145/800135.804414.

43 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986. doi:10.1109/SFCS.1986.25.

Interval Selection in Data Streams: Weighted Intervals and the Insertion-Deletion Setting

Jacques Dark
Unaffiliated Researcher, Cambridge, UK

Adithya Diddapur
School of Computer Science, University of Bristol, UK

Christian Konrad
School of Computer Science, University of Bristol, UK

Abstract

We study the Interval Selection problem in data streams: Given a stream of n intervals on the line, the objective is to compute a largest possible subset of non-overlapping intervals using $O(|OPT|)$ space, where $|OPT|$ is the size of an optimal solution. Previous work gave a 3/2-approximation for unit-length and a 2-approximation for arbitrary-length intervals [Emek et al., ICALP'12]. We extend this line of work to weighted intervals as well as to insertion-deletion streams. Our results include:

1. When considering weighted intervals, a $(3/2+\epsilon)$-approximation can be achieved for unit intervals, but any constant factor approximation for arbitrary-length intervals requires space $\Omega(n)$.
2. In the insertion-deletion setting where intervals can both be added and deleted, we prove that, even without weights, computing a constant factor approximation for arbitrary-length intervals requires space $\Omega(n)$, whereas in the weighted unit-length intervals case a $(2+\epsilon)$-approximation can be obtained.

Our lower bound results are obtained via reductions to the recently introduced Chained-Index communication problem, further demonstrating the strength of this problem in the context of streaming geometric independent set problems.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Streaming models

Keywords and phrases Streaming Algorithms, Interval Selection, Weighted Intervals, Insertion-deletion Streams

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.24

Funding *Jacques Dark*: Part of the work of J.D. was done while at the University of Warwick, supported by an EMEA Microsoft Research PhD studentship and European Research Council grant ERC-2014-CoG 647557.
Christian Konrad: Supported by EPSRC New Investigator Award EP/V010611/1.

1 Introduction

In this work, we study *streaming algorithms* for the Interval Selection problem. Streaming algorithms are characterized by processing the input data sequentially in a single pass while maintaining only a small sketch of the input data in memory. The objective is to design algorithms that use much less space than the size of the input stream. Due to the low memory requirements, streaming algorithms are well-suited for processing large data sets and constitute a well-established class of resource-constraint algorithms within algorithms research today.

Given a set $\mathcal{S}$ of n intervals on the line, the Interval Selection problem consists of finding a largest subset $T \subseteq \mathcal{S}$ of pairwise non-overlapping intervals and can be regarded as the Maximum Independent Set problem on *interval graphs*, i.e., the intersection graphs of intervals on the line. In the streaming model, we assume that the intervals $\mathcal{S}$ arrive in arbitrary

© Jacques Dark, Adithya Diddapur, and Christian Konrad;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 24; pp. 24:1–24:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

order (for example, in the form of tuples indicating their left and right boundaries). We distinguish between unit-length/arbitrary-length intervals and weighted/unweighted intervals. We will also consider both insertion-only and insertion-deletion streams, where, in the latter, previously inserted intervals may eventually be deleted.

Emek et al. [7] initiated the study of the Interval Selection problem in data streams. They gave a $\frac{3}{2}$-approximation algorithm for unit-length intervals and a 2-approximation algorithm for arbitrary-length intervals, with both algorithms using space $O(|OPT|)$, where OPT denotes an optimal solution (see also [3] for significantly simpler algorithms with the same quality guarantees). As proved by Emek et al., both results are optimal in that computing slightly better approximations requires space $\Omega(n)$. Bakshi et al. [1] were the first to consider weighted intervals and insertion-deletion streams, albeit through a slightly different lens: Their objective was to approximate the total weight of a heaviest independent set of intervals rather than outputting the intervals themselves. They prove that the weight of a heaviest independent set of weighted unit-length intervals in insertion-deletion streams can be approximated within a factor $2+\epsilon$ with only $\text{poly}(\frac{\log n}{\epsilon})$ space. Curiously, neither weighted intervals nor insertion-deletion streams have previously been considered in the setting where intervals need to be reported.

1.1 Our Results

In this work, we address this shortcoming and consider both weighted intervals and insertion-deletion streams. Without loss of generality, if weights are present then we assume that the maximum weight is W and the minimum weight is 1. Similarly, in the arbitrary-length case, we assume that the longest interval is of length Δ and the shortest interval is of length 1. Other settings can be obtained by scaling. We also mention at this stage that we assume that any interval (weighted or unweighted) can be stored in space $O(1)$ in order to avoid the complexities arising from how interval boundaries and weights are encoded. If instead space t is accounted for storing an interval then the space complexities of our algorithms need to be multiplied by t.

We observe that either a particular setting can be solved well in that a constant factor approximation can be achieved with space linear in $|OPT|$, or, a lower bound can be established, demonstrating that space $\Omega(n)$ is necessary for a constant factor approximation in the worst case, i.e., if W or Δ is large enough.

Insertion-only Streams. As our point of departure, we recall that, in insertion-only streams, $O(1)$-approximations for unweighted intervals can be achieved for both unit-length and arbitrary-length intervals with space $O(|OPT|)$ [7]. As our first set of results, we show that, when adding weights to these settings, then the unit-length interval case can still be solved well, however, in the arbitrary-length case, space $\Omega(n)$ is needed in the worst case:

▶ **Theorem 1.** *For every $\epsilon > 0$, there is a deterministic one-pass insertion-only $(\frac{3}{2}+\epsilon)$-approximation streaming algorithm for Interval Selection with space $O(\frac{\log(1/\epsilon)}{\epsilon}|OPT|)$ on weighted unit-length intervals.*

▶ **Theorem 2.** *Any one-pass constant error randomized streaming algorithm with approximation factor α for Interval Selection on weighted arbitrary-length intervals in insertion-only streams must use $\Omega\left(\frac{1}{\alpha^2}\min\{\Delta^{\frac{1}{2\alpha}}, \frac{n}{2^{2\alpha}}\}\right)$ bits of space.*

We observe that if $\Delta = \text{poly}(n)$ then the previous lower bound shows that space $\Omega(n)$ is needed to achieve a constant factor approximation.

Insertion-deletion Streams. We first observe that, in insertion-deletion streams, space as a function of $|OPT|$ cannot be achieved. To this end, consider an input stream consisting of $\Theta(n)$ disjoint intervals that are subsequently either deleted or not, each with probability $\frac{1}{2}$. On such an input distribution, any competitive algorithm must store most of these intervals since the algorithm cannot know whether the intervals will subsequently be deleted. Observe that, if the intervals are indeed deleted, then we have $|OPT| = 0$, while every constant factor approximation algorithm still requires space $\Omega(n)$. To circumvent this issue, we express the space complexities of our insertion-deletion algorithms in terms of $|OPT^*|$, where OPT^* is defined as the largest independent set of intervals *on any prefix of the input stream.* Note that, even in the weighted case, OPT^* is defined as the *largest* independent set on any prefix. While it appears natural to consider the heaviest independent set on any prefix instead, this quantity is not useful since, for example, one could prefix any input stream with the insertion and subsequent deletion of an arbitrarily heavy interval. In this case, the size of a heaviest independent set on any prefix of the stream would always be one.

As our second set of results, we show that unit-length intervals (both unweighted and weighted) can be solved well in insertion-deletion streams, but, unlike in the insertion-only model, constant factor approximations in the unweighted arbitrary-length case require space $\Omega(n)$ in the worst case.

▶ **Theorem 3.** *There is a randomized one-pass insertion-deletion 2-approximation streaming algorithm for* Interval Selection *on unweighted unit-length intervals with space $O(|OPT^*| \cdot \log^3 n)$ that succeeds with high probability. In the weighted case, a $(2+\epsilon)$-approximation can be achieved with space $O(|OPT^*| \cdot \log^3(n) \cdot \log(W)/\epsilon)$.*

A lower bound by Bakshi et al. [1] shows that the result in the previous theorem is best possible in that, even in the unweighted case, every $(2-\epsilon)$-approximation, for any $\epsilon > 0$, requires $\Omega(n)$ space.

▶ **Theorem 4.** *Any one-pass constant error randomized streaming algorithm with approximation factor α for* Interval Selection *on unweighted arbitrary-length intervals in insertion-deletion streams must use $\Omega\left(\frac{1}{\alpha^2}\min\{\Delta^{\frac{1}{\alpha^2}}, \frac{n}{\alpha}\}\right)$ bits of space.*

1.2 Techniques

We will first discuss the ideas behind our algorithms and then address our lower bounds.

Algorithms. To obtain our algorithms, we use an idea that goes back to Hochbaum and Mass [12] as it is applied to the Interval Selection problem by Cabello and Pérez-Lantero [3]. Cabello and Pérez-Lantero observed that, if an optimal algorithm for Interval Selection with space $O(1)$ on unit-length intervals that are all contained in the domain $[0, 3)$ can be obtained, then a $\frac{3}{2}$-approximation algorithm with space $O(|OPT|)$ can be achieved without any restrictions on where the intervals are located. We generalize this idea and show that if an α-approximation streaming algorithm with space s for unit length intervals that are located within the domain $[0, \gamma)$ exists, for some integer γ, then a $(\alpha \cdot \gamma/(\gamma - 1))$-approximation streaming algorithm with space $O(s \cdot \gamma \cdot |OPT|)$ without any restrictions on where the intervals are located can be obtained in insertion-only streams, and with space $O((s + \log n) \cdot \gamma \cdot |OPT^*|)$ in insertion-deletion streams. Our generalization works for both the unweighted and weighted cases.

We thus focus on designing streaming algorithms that perform well when all intervals in the input stream are contained in a small window $[0, \gamma)$, for some constant integer γ. In the insertion-only model, we give a $(1+\epsilon)$-approximation algorithm with space $O(\frac{\log(1/\epsilon)}{\epsilon})$ for

the domain $[0, 3)$ for weighted unit-length intervals that makes use of weight classes, which, using the idea mentioned above, translates to a $(\frac{3}{2} + \epsilon)$-approximation algorithms with space $O(\frac{\log(1/\epsilon)}{\epsilon} \cdot |OPT|)$ for intervals without restrictions on the domain. In the insertion-deletion setting, we give a $(1 + \epsilon)$-approximation streaming algorithm with space $O(\frac{\log W}{\epsilon} \cdot \log^3 n)$ for weighted unit-length intervals contained in the domain $[0, 2)$ that relies on ℓ_0-sampling and the use of weight classes, which translates to a $(2 + \epsilon)$-approximation for arbitrary input instances with space $O(\frac{\log W}{\epsilon} \cdot \log^3 n \cdot |OPT^*|)$.

Lower Bounds. Our lower bounds are reductions to the recently introduced one-way p-party communication problem $\mathsf{CHAIN}_p(k)$ [5] and to an extension that we introduce here that we denote by $\mathsf{AUG\text{-}CHAIN}_p(k)$.

In the *one-way p-party communication model* (see [14] for an introduction to the topic), p parties, who each hold a portion of the input, communicate in a one-way fashion, i.e., the first party sends a message to the second, who, upon receipt, sends a message to the third party. This continues until party p has received a message from party $p - 1$ and outputs the result of the computation. The objective is to design communication protocols such that the size of the largest message is as small as possible. The parties have access to both private and public randomness, and they are allowed to err with small constant probability. Streaming algorithms give rise to one-way communication protocols, and a lower bound on the size of a largest message in every one-way communication protocol for a problem P then implies a lower bound on the space required by streaming algorithms for problem P.

The problem $\mathsf{CHAIN}_p(k)$ can be regarded as chaining together $p - 1$ instances of the well-known INDEX_k problem. INDEX_k is a two-party communication game where the first party holds a bit string $X = X[0], \ldots, X[k - 1] \in \{0, 1\}^k$ and the second party holds an index $\sigma \in [k - 1] := \{0, \ldots, k - 1\}$. The first party sends a single message to the second who is tasked with outputting the bit $X[\sigma]$. It is known that randomized constant-error protocols that solve INDEX_k must send a message of size $\Omega(k)$. In $\mathsf{CHAIN}_p(k)$, each pair of subsequent parties $i, i + 1$, for $1 \leq i < p$, holds a separate INDEX_k instance (X^i, σ^i). The overall $p - 1$ instances are correlated in that

$$X^1[\sigma^1] = X^2[\sigma^2] = \cdots = X^{p-1}[\sigma^{p-1}] = z \in \{0, 1\} \ ,$$

and the objective for party p is to determine the value of z. It is known that a protocol that solves $\mathsf{CHAIN}_p(k)$ needs to send at least one message of size $\Omega(\frac{k}{p^2})$ [8].

The key idea of our lower bound construction for weighted arbitrary-length intervals in the insertion-only model is to encode each INDEX_k input (X^i, σ^i) in $\mathsf{CHAIN}_p(k)$ into one or multiple *clique gadgets*, i.e., a set of k slightly shifted but overlapping potential intervals where the jth interval is contained in the gadget if and only if$X^i[j] = 1$. We call the σ^ith potential interval *special*. The strength of the correlation of the outputs of all the INDEX_k instances in $\mathsf{CHAIN}_p(k)$ is that either *all* or *none* of the special intervals are contained in the constructed clique gadgets. Our construction is such that, if the special intervals are contained in the instance, then a heavy independent set containing the special intervals exists, and otherwise, no heavy independent set exists. We argue that any streaming algorithm that computes a good enough approximation to Interval Selection can be used as a one-way communication protocol that solves $\mathsf{CHAIN}_p(k)$, which establishes our lower bound.

Our lower bound for the insertion-deletion setting makes use of the $\mathsf{AUG\text{-}CHAIN}_p(k)$ problem that we introduce in this work. $\mathsf{AUG\text{-}CHAIN}_p(k)$ differs from $\mathsf{CHAIN}_p(k)$ in that, instead of chaining INDEX_k instances together, we chain $\mathsf{AUGMENTED\text{-}INDEX}_k$ instances together. The $\mathsf{AUGMENTED\text{-}INDEX}_k$ problem is a slight modification of INDEX_k where,

besides the index σ, Bob also holds the prefix $X[0], \ldots, X[\sigma - 1]$ of X. The lower bound established by Feldman et al. for $\mathsf{CHAIN}_p(k)$ carries over to $\mathsf{AUG\text{-}CHAIN}_p(k)$ essentially without modifications.

The $\mathsf{AUGMENTED\text{-}INDEX}_k$ problem (and thus also the $\mathsf{AUG\text{-}CHAIN}_p(k)$ problem) is well-suited for proving lower bounds in insertion-deletion streams. Since Bob knows the prefix $X[0], \ldots, X[\sigma - 1]$, Bob can delete the intervals of the clique gadget introduced by Alice that correspond to the prefix $X[0], \ldots, X[\sigma - 1]$. This provides an additional advantage over INDEX_k that allows us to obtain a lower bound for unweighted arbitrary-length intervals that could not be obtained without deletions. For details, please see Section 4.2.

1.3 Further Related Work

As previously mentioned, the Interval Selection problem is a special case of the Maximum Independent Set (MIS) problem, which has received significant attention in the streaming model. Despite the fact that MIS is NP-hard and hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$ [11], in the streaming setting, a one-pass α-approximation can be obtained by storing $O(\frac{n^2}{\alpha^2})$ edges, by sampling a random vertex-induced subgraph on $\Theta(\frac{n}{\alpha})$ vertices and computing a maximum independent set in this subgraph in exponential time [10] in the post-processing step. In the vertex-arrival order setting, where vertices arrive one-by-one together with their incident edges that connect to previously arrived vertices, Cormode et al. proved that space $\Omega(n^2/\alpha^6)$ is necessary for an α-approximation [5]. Furthermore, efficient streaming algorithms exist for computing independent sets that achieve the Caro-Wei bound, see [9] and [4].

The $\mathsf{CHAIN}_p(k)$ problem was introduced by Cormode et al. [5] in order to establish the lower bound mentioned above for vertex-arrival MIS. They also used $\mathsf{CHAIN}_3(k)$ to show that obtaining a better than 2.5-approximation streaming algorithm to the problem of computing a largest subset of a stream of non-overlapping axis-aligned unit squares requires space $\Omega(n)$. Bhore et al. [2] then also used $\mathsf{CHAIN}_p(k)$ to prove that any constant factor approximation algorithm to solve the interval selection problem on split intervals requires space $\Omega(n)$. Feldman et al. [8] used $\mathsf{CHAIN}_p(k)$ for proving lower bounds in the context of one-way communication protocols for sublinear maximization and gave the strongest known lower bound of $\Omega(n/p^2)$ on the size of at least one message in a protocol that solves the problem.

1.4 Outline

In Section 2, we discuss tools and known results that we make use of in the work. Then, in Section 3, we give our results that apply to the insertion-only setting, i.e., our algorithm for weighted unit-length intervals and our lower bound for weighted arbitrary-length intervals. Then, in Section 4, we give our results that address the insertion-deletion setting, namely, our algorithm for weighted unit-length intervals and our lower bound for unweighted arbitrary-length intervals. Finally, we conclude with open problems in Section 5.

2 Preliminaries

For an integer k, we define $[k] := \{0, 1, \ldots, k\}$.

Interval Selection. In the Interval Selection problem, we are given a set $\mathcal{S}$ of n (potentially weighted) intervals on the line. If weights are present then, for an interval $I \in \mathcal{S}$, we write $w(I)$ to denote its weight. In the presence of weights, the objective is to identify a

subset of pairwise non-overlapping intervals of maximum weight. We denote the maximum interval weight in an instance by W and assume that the smallest weight is 1. If no weights are present then the objective is to find a largest cardinality subset of non-overlapping intervals. We distinguish between unit-length intervals, where all intervals are of length 1, and arbitrary-length intervals. In the arbitrary-length case, we denote by Δ the length of the longest interval and assume that the shortest interval is of length 1.

Let OPT denote an independent set of intervals of maximum cardinality (in the unweighted case) or of maximum weight (in the weighted case) at the end of the stream. For insertion-deletion streams, we will also use OPT^*, which denotes an independent set of intervals of largest cardinality across all prefixes of the stream. Then, we denote the cardinality of these solutions by $|OPT|$ ($|OPT^*|$, respectively), and the weight by $w(OPT)$ ($w(OPT^*)$, respectively).

Streaming Models. In the insertion-only model, a streaming algorithm sees the n intervals in arbitrary order. In the insertion-deletion model, the input stream consists of a sequence of interval insertions and deletion such that, overall, at most n different intervals have been inserted. We assume that only intervals that have previously been inserted can be deleted.

We assume that storing an interval together with its left and right boundaries as well as its weight requires space $O(1)$. This assumption is for simplicity since the focus of our work is not on how to best encode intervals. We remark, however, that if space t is accounted for storing an interval then the space complexities of our algorithms increase by a factor of t.

An important technique for designing insertion-deletion streaming algorithms is ℓ_0-sampling (see, for example, [6]) that we will also use in this work.

ℓ_0-Sampling. Let $a \in \mathbb{N}_0^n$ be a non-zero vector. The objective in ℓ_0-sampling is to sample a uniform random non-zero coordinate from a. An insertion-deletion stream can be described as a sequence of increases/decreases of the coordinates of an underlying n-dimensional vector. Applied to the Interval Selection problem, each coordinate corresponds to an interval in the input stream.

In this work we use the ℓ_0-sampler by Jowhari et al. [13].

▶ **Theorem 5** ([13]). *There exists an insertion-deletion streaming algorithm that performs ℓ_0-sampling which uses space $O(\log^2 n \log(1/\delta))$ and succeeds with probability at least $1-\delta$.*

Communication Problems. In this paper, we work with the $\mathsf{CHAIN_p(k)}$ problem introduced by Cormode et al. [5].

▶ **Definition 6.** *$\mathsf{CHAIN_p(k)}$ is a one-way p-party communication problem where the players hold parts of the input as follows:*

- *The first player p_1 holds a vector $X^1 \in \{0,1\}^k$.*
- *For each $2 \le i \le (p-1)$, player p_i holds a vector $X^i \in \{0,1\}^k$ and an index $\sigma^{i-1} \in [k-1]$.*
- *The last player p_p holds an index $\sigma^{p-1} \in [k-1]$.*

We are guaranteed that there exists an answer bit $z \in \{0,1\}$ such that $X^i[\sigma^i] = z$ for all $1 \le i \le p-1$. It will be convenient to denote a specific instance of $\mathsf{CHAIN_p(k)}$ by $(X^1, \sigma^1, \ldots, X^{p-1}, \sigma^{p-1})$.

The game proceeds with the players sequentially taking turns to send a single message to the next player i.e. p_1 sends a message m^1 to p_2 followed by p_2 sending m^2 to p_3 and so on. The objective of the game is for p_p to output z.

$\mathsf{CHAIN_p(k)}$ can be trivially solved with $O(k)$ bits of total communication: Any of the parties p_i sends X^i to party p_{i+1}. Feldman et al. [8] proved the following lower bound:

▶ **Theorem 7** ([8]). *For any positive integers k and $p \geq 2$, any (potentially randomized) protocol for $\mathsf{CHAIN_p(k)}$ with success probability at least $2/3$ must send a message of size $\Omega(k/p^2)$.*

We now define the related multi-party one-way communication problem $\mathsf{AUG\text{-}CHAIN_p(k)}$.

▶ **Definition 8.** *$\mathsf{AUG\text{-}CHAIN_p(k)}$ is a p-party one-way communication problem where:*
- *The first player holds a vector $X^1 \in \{0,1\}^k$.*
- *For each $2 \leq i \leq (p-1)$, p_i holds a vector $X^i \in \{0,1\}^k$, an index $\sigma^{i-1} \in [k-1]$, and the prefix $X_+^{i-1} = (X^{i-1}[0], \ldots, X^{i-1}[\sigma^{i-1}-1])$.*
- *The last player p_p holds an index $\sigma^{p-1} \in [k-1]$ and the prefix $X_+^{p-1} = (X^{p-1}[0], \ldots, X^{p-1}[\sigma^{p-1}-1])$.*

Again we are guaranteed that there exists an answer bit $z \in \{0,1\}$ such that $X^i[\sigma^i] = z$ for all $1 \leq i \leq p-1$, and we denote a specific instance by $(X^1, \sigma^1, \ldots, X^{p-1}, \sigma^{p-1})$. The game then proceeds the same as for $\mathsf{CHAIN_p(k)}$.

It is not hard to see that the analysis by Feldman et al. for $\mathsf{CHAIN_p(k)}$ also holds for $\mathsf{AUG\text{-}CHAIN_p(k)}$ with essentially no modification. We thus have:

▶ **Theorem 9.** *For any positive integers k and $p \geq 2$, any (potentially randomized) protocol for $\mathsf{AUG\text{-}CHAIN_p(k)}$ with success probability at least $2/3$ must send a message of size at least $\Omega(k/p^2)$.*

3 Insertion-only Streams

In this section, we give our $(\frac{3}{2}+\epsilon)$-approximation streaming algorithm for weighted unit-length intervals. To this end, we first show in Subsection 3.1 that if an algorithm can be designed for unit intervals that arrive in the restricted domain $[0,\gamma)$, for some integer γ, then an algorithm with slightly increased approximation guarantee can be obtained without any restrictions on the domain of the intervals. Then, we present our algorithm in Subsection 3.2.

3.1 From Small Domains to Unrestricted Domains

We begin with notation for partitioning the real line into windows of integral length γ. We define a single window of length γ located at $i \in \mathbb{R}$ as $W_i^\gamma = [i, i+\gamma)$. We then define a partition of the real line as $\mathbb{W}_i^\gamma = \{W_{i+j\cdot\gamma}^\gamma : j \in \mathbb{Z}\}$.

▶ **Lemma 10.** *Let $\mathcal{A}$ be a one-pass insertion-only (insertion-deletion) streaming algorithm with approximation factor α and space $O(s)$ on unit-length weighted intervals such that all intervals are fully contained within the restricted domain $W_0^\gamma = [0,\gamma)$, for some integer $\gamma \geq 2$. Then there exists a one-pass insertion-only (respectively insertion-deletion) streaming algorithm $\mathcal{A}'$ with approximation factor $\alpha \cdot \gamma/(\gamma-1)$ and space $O(s \cdot \gamma \cdot |OPT|)$ (respectively $O((s+\log n)\cdot\gamma\cdot|OPT^*|)$) for unit interval streams with unrestricted interval domains.*

Proof. To extend $\mathcal{A}$ to $\mathcal{A}'$ we make use of γ partitions of the real line $\mathbb{W}_0^\gamma, \ldots, \mathbb{W}_{\gamma-1}^\gamma$. Let I be an interval which arrives in the stream and let $\mathcal{W}$ be the set of windows across all partitions which fully contain I. Then, for each $w \in \mathcal{W}$, if w does not fully contain any existing intervals at that point in the stream then $\mathcal{A}'$ creates a new instance of $\mathcal{A}$ in w. Then, for each $w \in \mathcal{W}$, I is passed as an input to the instance of $\mathcal{A}$ associated with w, translated

so that it appears to be contained in W_0^γ from the perspective of the instance of $\mathcal{A}$. We denote a window as active if it contains at least one interval and therefore has a running instance of $\mathcal{A}$. In the insertion-deletion setting, each active window also maintains a counter of the number of intervals that it currently contains, and if this counter drops to 0 then $\mathcal{A}'$ clears the corresponding instance of $\mathcal{A}$ and marks the window as inactive. Since the windows within a partition $\mathbb{W}_a^\gamma$ are pairwise disjoint, taking the union of independent sets of intervals across the windows in a partition will also yield an independent set, which we denote as ALG_a. At the end of the stream, $\mathcal{A}'$ outputs the largest ALG_a across $a \in \{0, \ldots, \gamma - 1\}$.

Approximation Factor. We begin with the approximation factor of $\mathcal{A}'$. Let OPT be a heaviest independent set, and for each $0 \leq a \leq \gamma - 1$, let OPT_a be the set of intervals in OPT which are fully contained in the windows of $\mathbb{W}_a^\gamma$.

First, observe that every interval is contained in precisely $\gamma - 1$ windows. Thus, applying this observation to OPT, we obtain

$$\sum_{a=0}^{\gamma-1} w(OPT_a) = (\gamma - 1)w(OPT) \ .$$

Let j be such that $w(OPT_j) = \max\{w(OPT_a) : a \in \{0, \ldots, \gamma - 1\}\}$. Then, from the previous equation, we obtain

$$w(OPT) \leq \frac{\gamma}{\gamma - 1} \cdot w(OPT_j).$$

Next, since $\mathcal{A}$ returns an α-approximation on each active window and they are disjoint within a given partition $\mathbb{W}_j^\gamma$, we get that $w(OPT_j) \leq \alpha \cdot w(ALG_j)$, and so

$$w(OPT) \leq \frac{\gamma}{\gamma - 1} w(OPT_j) \leq \frac{\gamma}{\gamma - 1} \cdot \alpha \cdot w(ALG_j) \ ,$$

which establishes the approximation factor of $\mathcal{A}'$.

Space Requirements. We first argue the space bound in the insertion-only setting. As previously mentioned, every interval is contained in exactly $\gamma - 1$ windows across all the window partitions. Hence, the intervals of OPT are contained in at most $(\gamma - 1) \cdot |OPT|$ different windows. Denote this set of windows by $\mathcal{W}_{OPT}$. Then, based on $\mathcal{W}_{OPT}$, we define the set of windows $\mathcal{W}' \supset \mathcal{W}_{OPT}$ as follows: For each window $[a, a + \gamma) \in \mathcal{W}_{OPT}$, add the two windows $[a - 1, a - 1 + \gamma)$ and $[a + 1, a + 1 + \gamma)$ to $\mathcal{W}'$. Then, observe that $|\mathcal{W}'| \leq 3 \cdot |\mathcal{W}|$.

We will now argue that, for every interval I in the input stream, the $\gamma - 1$ windows that I is contained in are also in $\mathcal{W}'$. Indeed, observe that, since OPT is also a maximal independent set, I intersects with an interval $J \in OPT$, and assume that I intersects J on J's left boundary (the case where I intersects on the right boundary is similar and omitted). Consider now all the $\gamma - 1$ windows that contain J. Then, for each of these windows, either I is also contained in it or it is contained in the window shifted by 1 to the left. Since we added these shifted windows to $\mathcal{W}'$, it follows that all of the $\gamma - 1$ windows that contain I are thus also contained in $\mathcal{W}'$. Hence, $\mathcal{A}'$ executes at most $|\mathcal{W}'| \leq 3 \cdot (\gamma - 1) \cdot |OPT|$ instances of $\mathcal{A}$, and since each instance requires space s, we conclude that $\mathcal{A}$ uses space $O(s \cdot \gamma \cdot |OPT|)$, which establishes the space bound of the algorithm in the insertion-only setting.

Consider now the insertion-deletion setting. Consider any moment during the processing of the stream, and let OPT' be an independent set of maximum cardinality at this moment. For the exact same reasoning as above, there are $O(\gamma \cdot |OPT'|)$ instances of $\mathcal{A}$ simultaneously running at this moment. Observe that, in addition to the memory state of $\mathcal{A}$, for each

simultaneous run, the algorithm $\mathcal{A}'$ also maintains a counter of the number of intervals that currently exist in the respective window. The space complexity in the insertion-deletion setting is thus $O(\gamma \cdot |OPT'| \cdot (s + \log n))$. The result follows since $|OPT'| \leq |OPT^*|$ holds. ◀

3.2 Algorithm for Weighted Unit-length Intervals

Equipped with Lemma 10, we are now ready to prove the following result:

▶ **Theorem 1.** *For every $\epsilon > 0$, there is a deterministic one-pass insertion-only $(\frac{3}{2} + \epsilon)$-approximation streaming algorithm for Interval Selection with space $O(\frac{\log(1/\epsilon)}{\epsilon}|OPT|)$ on weighted unit-length intervals.*

Theorem 1 is an immediate consequence of Lemma 10 and Algorithm 1, which is analyzed in Lemma 11.

Algorithm 1 is a $(1 + \epsilon)$-approximation algorithm for the weighted unit-length case when all intervals are included in the domain $[0, 3)$. The algorithm maintains the heaviest interval $I_{\max}$ observed so far and would only retain further intervals of weight at least $\epsilon \cdot w(I_{\max})$. The algorithm uses exponentially increasing weight classes $[(1 + \epsilon)^i, (1 + \epsilon)^{i+1})$, and for each weight class, stores the leftmost and rightmost intervals that fall within the respective weight class, where leftmost and rightmost are defined in the obvious way. We prove in Lemma 11 that the heaviest independent set that can be formed from the intervals stored constitutes a $(1 + \epsilon)$-approximation.

Algorithm 1 $(1 + \epsilon)$-approximate one-pass streaming algorithm for weighted unit-length intervals on the restricted domain $[0, 3)$.

Require: An insertion-only stream of weighted intervals contained within the window $[0, 3)$
1: $I_{\max} \leftarrow \perp$ (we assume that $w(\perp) = -\infty$){Maintain the heaviest interval}
2: **for** each interval I in the input stream **do**
3: **if** $w(I) > w(I_{\max})$ **then**
4: $I_{\max} \leftarrow I$
5: remove all intervals stored with weight less than $\epsilon \cdot w(I_{\max})$
6: **else if** $w(I) < \epsilon \cdot w(I_{\max})$ **then**
7: **continue** with next interval in stream
8: **end if**
9: Let j be the weight class of I, i.e., let j be such that $(1 + \epsilon)^{j-1} \leq w(I) < (1 + \epsilon)^j$
10: Let S_j be the set of (at most two) intervals stored from weight class j
11: $S_j \leftarrow$ left-most and right-most intervals among $S_j \cup \{I\}$
12: **end for**
13: **return** Heaviest independent set among all intervals stored

▶ **Lemma 11.** *Algorithm 1 is a deterministic one-pass $(1 + \epsilon)$-approximation streaming algorithm for Interval Selection with space $O(\frac{\log(1/\epsilon)}{\epsilon})$ on weighted unit-length intervals that are restricted to the domain $[0, 3)$.*

Proof. We first establish the space required by the algorithm. Consider any moment during the processing of the stream and let $I_{\max}$ be the heaviest interval. Let j^* denote the weight class that contains $I_{\max}$, i.e., j^* is such that $(1+\epsilon)^{j^*-1} \leq w(I_{\max}) < (1+\epsilon)^{j^*}$. The algorithm only stores intervals of weight at least $\epsilon \cdot w(I_{\max})$, and at most two intervals per weight class. Recall that the maximum weight in weight class j is $(1 + \epsilon)^j$. Then, if $(1 + \epsilon)^j < \epsilon(1 + \epsilon)^{j^*-1}$ then intervals in weight class j are not stored. This implies that, if intervals in a weight class j are stored, then

$$(1+\epsilon)^j \geq \epsilon(1+\epsilon)^{j^*-1} \text{ , which implies}$$
$$j^* - j = O\left(\frac{\log(1/\epsilon)}{\epsilon}\right) .$$

Hence, there are only $O\left(\frac{\log(1/\epsilon)}{\epsilon}\right)$ weight classes where intervals are stored. This establishes the space complexity of the algorithm.

Regarding the approximation factor, let OPT denote an optimal solution. If $|OPT| = 1$ then our algorithm finds an optimal solution since the algorithm stores the heaviest interval. Hence, assume that $|OPT| = 2$. Suppose first that at least one of the two intervals is of weight at most $\epsilon \cdot w(I_{\max})$. Then, the interval $I_{\max}$ yields a $(1+\epsilon)$-approximation. We can thus assume that both intervals in OPT are of weight at least $\epsilon \cdot w(I_{\max})$. Let $OPT = \{I_L, I_R\}$, and assume that I_L is located to the left of I_R. Furthermore, let j_L denote the weight class of I_L and j_R denote the weight class of I_R. Then, let I'_L be the left-most interval stored by the algorithm in weight class j_L, and let I'_R be the right-most interval stored in weight class j_R. Observe that these intervals exist since I_L and I_R also arrived in the stream, and the algorithm either stores these intervals or different intervals of these weight classes that are located further to the left of I_L or to the right of I_R. Since I_L and I'_L are in the same weight class, and since I_R and I'_R are in the same weight class, we obtain:

$$w(OPT) = w(I_L) + w(I_R) \leq (1+\epsilon)(w(I'_L) + w(I'_R)) .$$

This completes the proof. ◀

We remark that the space required by our algorithm is independent of W, the maximum weight of an interval.

Our main theorem of this section then follows from plugging in the algorithm from Lemma 11 into Lemma 10.

3.3 Lower Bound for Weighted Arbitrary-length Intervals

In this section, we will prove our lower bound for weighted arbitrary-length intervals in insertion-only streams.

▶ **Theorem 2.** *Any one-pass constant error randomized streaming algorithm with approximation factor α for* Interval Selection *on weighted arbitrary-length intervals in insertion-only streams must use $\Omega\left(\frac{1}{\alpha^2}\min\{\Delta^{\frac{1}{2\alpha}}, \frac{n}{2^{2\alpha}}\}\right)$ bits of space.*

Our construction relies on clique gadgets that we will define in Subsection 3.3.1. Then, we show that a protocol for $\mathsf{CHAIN}_p(k)$ can be obtained from an algorithm for Interval Selection in Subsection 3.3.2. Finally, we give the analysis in Subsection 3.3.3.

3.3.1 Clique Gadgets

We now give our definition of clique gadgets and provide an illustration in Figure 1.

▶ **Definition 12** (Clique Gadget)**.** *Let $X \in \{0,1\}^k$ be a binary vector, let $[a,b] \subset \mathbb{R}$ be an interval with $a < b$, and define the* offset $s_{a,b} = \frac{b-a}{2(k+1)}$. *Then, we define an* unweighted clique gadget *with k intervals as*

$$\mathcal{K}(X, k, [a,b]) = \{I_i : 0 \leq i < k, X[i] = 1\} \text{ , where}$$
$$I_i = \left[a + (i+2)\cdot s_{a,b}, \frac{a+b}{2} + i \cdot s_{a,b}\right] .$$

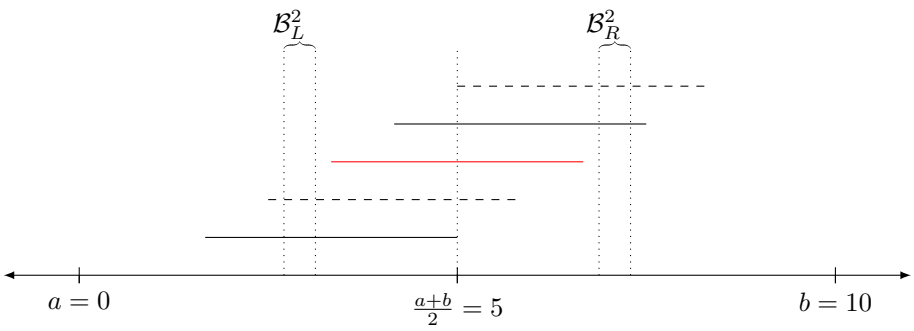


Figure 1 Example clique gadget $\mathcal{K}(X, k, [a, b])$ for $X = (1, 0, 1, 1, 0)$, $k = 5$, $[a, b] = [0, 10]$, with interval 2 shown in red. The corresponding left and right border regions $\mathcal{B}_L^2$ and $\mathcal{B}_R^2$ are also highlighted noting $|\mathcal{B}_L^2| = |\mathcal{B}_R^2| = s_{a,b}/2 = 5/12$.

We refer to the intervals I_i as potential intervals*, and a potential interval I_i is also an actual interval if $X[i] = 1$. For each $0 \leq i < k$, we define the left and right border regions of I_i by*

$$\mathcal{B}_L^i = \left[a + \left(i + \frac{5}{4}\right) \cdot s_{a,b}, a + \left(i + \frac{7}{4}\right) \cdot s_{a,b}\right] ,$$
$$\mathcal{B}_R^i = \left[\frac{a+b}{2} + \left(i + \frac{1}{4}\right) \cdot s_{a,b}, \frac{a+b}{2} + \left(i + \frac{3}{4}\right) \cdot s_{a,b}\right] ,$$

Last, we define a weighted clique gadget $\mathcal{K}(X, k, [a, b], w)$ as the unweighted clique gadget $\mathcal{K}(X, k, [a, b])$ where each interval has weight $w \in \mathbb{R}$.

Clique gadgets have the following properties:

▶ **Observation 13.** *The clique gadget $\mathcal{K}(X, k, [a, b])$ has the following properties:*

1. *All intervals and border regions are fully contained within the set (a, b).*
2. *All intervals contain the midpoint $m = \frac{b+a}{2}$ and thus form a clique.*
3. *For every i, the border regions have width $|\mathcal{B}_L^i| = |\mathcal{B}_R^i| = s_{a,b}/2 > 0$.*
4. *For every i, every point in $\mathcal{B}_L^i$ intersects with every potential interval I_j where $j < i$.*
5. *For every i, every point in $\mathcal{B}_R^i$ intersects with every potential interval I_j where $i < j$.*
6. *For every i, $(\mathcal{B}_L^i \cup \mathcal{B}_R^i) \cap I_i = \emptyset$.*

3.3.2 Reduction

We prove our lower bound via a reduction to the $\mathsf{CHAIN}_p(k)$ problem. To this end, we will show that a one-pass streaming algorithm $\mathcal{A}$ for Interval Selection for weighted arbitrary-length intervals can be used to obtain a protocol π_{Chain} that solves $\mathsf{CHAIN}_p(k)$.

The reduction is described in Figure 2.

Construction of π_{Chain}:

For parameters p and k, let $(X^1, \sigma^1, \ldots, X^{p-1}, \sigma^{p-1})$ be a $\mathsf{CHAIN}_p(k)$ instance.

The parties base their construction on positive reals a, b, w with $a < b$ whose values we determine later. The parties proceed as follows:

- **Party 1:** Based on input X^1, party 1 constructs the clique gadget $\mathcal{K}(X^1, k, [a, b], w)$, runs $\mathcal{A}$ on the created intervals, and send the memory state of $\mathcal{A}$ to party 2.
- **Party $1 < i \leq p$:** Denote by $\mathcal{K}_1, \ldots, \mathcal{K}_{2^{i-2}}$ the 2^{i-2} clique gadgets introduced by party $i-1$, and let $B_{L,j}$ and $B_{R,j}$ denote the left and right border regions of the σ^{i-1}th potential interval in $\mathcal{K}_j$. For each $j \in \{1, \ldots, 2^{i-2}\}$, if $i < p$ then party i introduces the clique gadgets $\mathcal{K}(X^i, k, B_{L,j}, \frac{w}{2^{i-1}})$ and $\mathcal{K}(X^i, k, B_{R,j}, \frac{w}{2^{i-1}})$, and if $i = p$ then party $i = p$ introduces two arbitrary intervals of weight $\frac{w}{2^{p-1}}$ that are fully contained in $B_{L,j}$ and $B_{R,j}$, recalling there is no vector X^p in a $\mathsf{CHAIN}_p(k)$ instance.
  Party i receives the memory state of an execution of $\mathcal{A}$ from party $i-1$, and then continues the execution of $\mathcal{A}$ on the intervals created, and if $i < p$, forwards the memory state to the next party. If $i = p$, then, as we will detail further below, party p is able to solve $\mathsf{CHAIN}_p(k)$ based on the output of $\mathcal{A}$.

Figure 2 Construction of π_{Chain} for solving $\mathsf{CHAIN}_p(k)$ given an algorithm $\mathcal{A}$ for Interval Selection on weighted unit-length intervals.

We observe that the clique gadgets created by the p parties can be represented as a rooted binary tree $\mathcal{T}$, where the first party's gadget constitutes the root node. For party $1 < i < p$, let $\mathcal{K}$ be a gadget inserted by party $i-1$, $\mathcal{K}_L$ and $\mathcal{K}_R$ the two gadgets inserted by party i into the left and right border regions of the special interval in $\mathcal{K}$. Then $\mathcal{K}_L$ and $\mathcal{K}_R$ are the two children of $\mathcal{K}$ in $\mathcal{T}$. The last party's intervals constitute the leaves of the tree. We will use $\mathcal{T}$ to determine the sizes of maximum independent sets when the underlying $\mathsf{CHAIN}_p(k)$ instance evaluates to either 1 or 0.

In this reduction, party i introduces 2^{i-1} clique gadgets and therefore at most $k \cdot 2^{i-1}$ intervals. The total number of intervals n is thus bounded by $n \leq \sum_{i=1}^{p} k \cdot 2^{i-1} < k \cdot 2^p$.

3.3.3 Analysis of the Reduction

We will argue in Lemma 14 that, depending on whether the answer bit in the $\mathsf{CHAIN}_p(k)$ instance is 1 or 0, the weight of a heaviest independent set is very different. Then, we conclude the section with the proof of our main result, Theorem 2.

▶ **Lemma 14.** *If the answer bit in the $\mathsf{CHAIN}_p(k)$ instance is 1 then the Interval Selection instance constructed in the reduction has an independent set of weight $w \cdot p$. If the answer bit is 0 then every independent set has weight at most $2 \cdot w$.*

Proof. The properties we use here to determine the independence of intervals from different gadgets follow from Observation 13.

Suppose first that the answer bit is 1. Then, for each party $1 \leq i < p$ and for each clique gadget constructed by party i, we pick the σ^ith interval of the gadget and add it to an independent set. We also pick all intervals created by party p. Observe that the resulting independent set corresponds to picking one interval from each node in $\mathcal{T}$. Overall, this creates an independent set of weight

$$\sum_{i=1}^{p} \frac{2^{i-1} \cdot w}{2^{i-1}} = w \cdot p \ ,$$

since party i holds 2^{i-1} clique gadgets where each interval has weight $\frac{w}{2^{i-1}}$.

Suppose now that the answer bit is 0. We argue via the tree structure $\mathcal{T}$ that every independent set is of weight at most $2 \cdot w$. To this end, we observe that, by construction of the instance, if an interval I from a clique gadget $\mathcal{K}$ is selected then I overlaps with all intervals in either the left or the right subtree of $\mathcal{K}$. Hence, since I is selected, none of the intervals in either the left or the right subtree of $\mathcal{K}$ can also be selected.

Consider now any independent set of intervals $\mathcal{I}$, and let $\mathcal{I}'$ be the independent set consisting only of party p's intervals. We will now argue that

$$w(\mathcal{I}) \leq 2 \cdot w(\mathcal{I}') \ , \tag{1}$$

and, since $w(\mathcal{I}') = w$, the result then follows.

To see why Inequality 1 holds, we conduct local transformations to turn $\mathcal{I}$ into $\mathcal{I}'$. To this end, repeat the following process: for each interval $I \in \mathcal{I}$ that is contained in a clique gadget $\mathcal{K}$ held by party $i < p$, we remove I from $\mathcal{I}$. Let $\mathcal{K}'$ be the child of $\mathcal{K}$ that does not contribute any intervals to $\mathcal{I}$. As previously argued, $\mathcal{K}'$ must exist. Then, we add back all of party p's intervals that are contained in $\mathcal{K}'$. When this process terminates then we have transformed $\mathcal{I}$ into $\mathcal{I}'$. Observe that, when removing I, we subtracted a weight of $w(I) = \frac{w}{2^{i-1}}$ from $\mathcal{I}$ and subsequently added 2^{p-i-1} intervals, each of weight $\frac{w}{2^{p-1}}$, back into $\mathcal{I}$, which corresponds to a total weight of $2^{p-i-1} \cdot \frac{w}{2^{p-1}} = \frac{w}{2^i} = w(I)/2$. Hence, overall, at most half of the initial weight of $\mathcal{I}$ is lost in this transformation, which proves Inequality 1 and completes the proof. ◀

Proof of Theorem 2. From Lemma 14, we obtain that if the answer bit in the $\mathsf{CHAIN}_p(k)$ instance is 1 then the instance has an independent set of weight $w \cdot p$, and if the answer bit is 0 then every independent set in the instance is of weight at most $2 \cdot w$. Hence, in our reduction, by the output of $\mathcal{A}$, party p can distinguish the two cases as long as the approximation factor α of $\mathcal{A}$ is such that $\alpha < \frac{p}{2}$. Theorem 7 thus yields that algorithm $\mathcal{A}$ must use space $\Omega(\frac{k}{p^2})$. It remains to pick suitable values for k and p. We pick $p = 2\alpha + 1$. We observe that the ratio between the longest and shortest intervals in our construction is $\Delta = (4(k+1))^{p-1} = (4(k+1))^{2\alpha}$, which implies that $k = \frac{1}{4}\Delta^{\frac{1}{2\alpha}} - 1$. Our lower bound thus becomes $\Omega(\Delta^{\frac{1}{2\alpha}}/\alpha^2)$. Since our construction uses $n \leq k \cdot 2^p = (\frac{1}{4}\Delta^{\frac{1}{2\alpha}} - 1) \cdot 2^{2\alpha+1} \leq \Delta^{\frac{1}{2\alpha}} \cdot 2^{2\alpha}$ intervals, our lower bound can never exceed the bound $O(\frac{n}{2^{2\alpha} \cdot \alpha^2})$.

Last, the construction departs with arbitrary reals a, b, w that determine the domain of the construction and the heaviest weight. We pick any a and select b large enough such that the shortest interval is of length 1, and we pick w large enough such that the lightest interval is of weight 1. This completes the proof. ◀

4 Insertion-deletion Streams

4.1 Algorithm for Unit-length Intervals

In this section, we give our insertion-deletion streaming algorithms for Interval Selection for both weighted and unweighted unit-length intervals. We prove the following theorem:

▶ **Theorem 3.** *There is a randomized one-pass insertion-deletion 2-approximation streaming algorithm for* Interval Selection *on unweighted unit-length intervals with space* $O(|OPT^*| \cdot \log^3 n)$ *that succeeds with high probability. In the weighted case, a* $(2+\epsilon)$*-approximation can be achieved with space* $O(|OPT^*| \cdot \log^3(n) \cdot \log(W)/\epsilon)$.

As in the insertion-only case, we argue in Lemma 15 that suitable algorithms for restricted domains exist and then apply Lemma 10 to obtain algorithms for unrestricted domains.

▶ **Lemma 15.** *There is a randomized one-pass insertion-deletion streaming algorithm with space* $O(\log^3 n)$ *for* Interval Selection *that produces an optimal solution for unweighted unit-length intervals that are restricted to the domain* $[0, 2)$. *For weighted intervals, the approximation factor becomes* $1+\epsilon$ *and the space is* $O(\frac{\log W}{\epsilon} \log^3 n)$.

Proof. We first consider the unweighted unit-length case. Since every pair of intervals that are contained in $[0, 2)$ intersect, the task reduces to outputting any interval if there is one. To achieve this task, we can run an ℓ_0-sampler that succeeds with high probability. To this end, we use the ℓ_0-sampler of Jowhari et al. (see Theorem 5) that uses space $O(\log^3 n)$ and errs with probability $\frac{1}{n^{10}}$.

Regarding weighted intervals, the task is to report an interval of weight at least $\frac{1}{1+\epsilon}$ times the weight of a heaviest interval. To this end, we run an ℓ_0-sampler for intervals in every weight class $[(1+\epsilon)^{j-1}, (1+\epsilon)^j)$, for every $j \in \{1, \ldots, \log_{1+\epsilon} W\}$. Observe that $\log_{1+\epsilon} W = O(\frac{\log W}{\epsilon})$ for small values of ϵ. The space complexity follows since each ℓ_o-sampler uses space $O(\log^3 n)$. By a union bound, the error probability of the resulting algorithm is at most $\frac{1}{n^{10}}$ times the number of weight classes that contain at least one interval, and since there are at most n such weight classes, the result follows. ◀

Proof of Theorem 3. The proof follows immediately from Lemma 15 and Lemma 10. ◀

4.2 Lower Bound for Unweighted Arbitrary-length Intervals

We will now prove our lower bound that applies to the unweighted, arbitrary-length setting in insertion-deletion streams:

▶ **Theorem 4.** *Any one-pass constant error randomized streaming algorithm with approximation factor* α *for* Interval Selection *on unweighted arbitrary-length intervals in insertion-deletion streams must use* $\Omega\left(\frac{1}{\alpha^2} \min\{\Delta^{\frac{1}{\alpha^2}}, \frac{n}{\alpha}\}\right)$ *bits of space.*

Similar to the lower bound of Section 3.3, we will give a reduction, however, instead of reducing to $\mathsf{CHAIN}_p(k)$, we reduce to $\mathsf{AUG\text{-}CHAIN}_p(k)$. Our construction will make use of the clique gadgets defined in Section 3.3. We will give our reduction in Section 4.2.1, however, due to space restrictions, the analysis is postponed to Appendix A.

4.2.1 Reduction

We will now show that a one-pass insertion-deletion streaming algorithm $\mathcal{A}$ for Interval Selection for unweighted arbitrary-length intervals can be used to obtain a protocol $\pi_{\text{Aug-Chain}}$ that solves $\mathsf{AUG\text{-}CHAIN}_p(k)$. The reduction is described in Figure 3.

Construction of $\pi_{\text{Aug-Chain}}$: For parameters p and k, let $(X^1, \sigma^1, \ldots, X^{p-1}, \sigma^{p-1})$ be a AUG-CHAIN$_p(k)$ instance. The parties base their construction on positive reals a, b with $a < b$ whose values we determine later. The parties proceed as follows:

- **Party 1:** Based on input X^1, party 1 constructs the unweighted clique gadget $\mathcal{K}(X^1, k, [a, b])$, runs $\mathcal{A}$ on the created intervals, and send the memory state of $\mathcal{A}$ to party 2.
- **Party $1 < i \leq p$:** Denote by $\mathcal{K}$ the clique gadget constructed by party $i-1$, and let B_R denote the right border region of the σ^{i-1}th potential interval in $\mathcal{K}$. Party i deletes all actual intervals of $\mathcal{K}$ with index below σ^{i-1}, which is possible since party i knows the prefix $X_+^{i-1} = \{X^{i-1}[0], \ldots, X^{i-1}[\sigma^{i-1} - 1]\}$. Then, if $i < p$, party i introduces the clique gadget $\mathcal{K}(X^i, k, B_R)$, and if $i = p$ then party $i = p$ introduces one arbitrary interval that is fully contained in B_R.
  Party i receives the memory state of an execution of $\mathcal{A}$ from party $i-1$, and then continues the execution of $\mathcal{A}$ on the intervals created, and if $i < p$, forwards the memory state to the next party. If $i = p$, then, as we will detail further below, party p is able to solve AUG-CHAIN$_p(k)$ based on the output of $\mathcal{A}$.

Figure 3 Construction of $\pi_{\text{Aug-Chain}}$ for solving AUG-CHAIN$_p(k)$ given an insertion-deletion streaming algorithm $\mathcal{A}$ for Interval Selection on unweighted unit-length intervals.

We observe that the number of intervals n in this construction is bounded by $n \leq k \cdot p$. For the analysis of this reduction, we refer the reader to Appendix A.

5 Conclusion

In this paper, we gave streaming algorithms for Interval Selection on weighted unit-length intervals in the insertion-only and the insertion-deletion settings. We also gave space lower bounds, one for the case of weighted arbitrary-length intervals in the insertion-only setting, and one for the case of unweighted arbitrary-length intervals in the insertion-deletion setting.

Our results demonstrate a jump in complexity when going from the insertion-only to the insertion-deletion setting: While a constant factor approximation for arbitrary-length intervals can be achieved in the insertion-only setting with space $O(|OPT|)$, space $\Omega(n)$ is required in the insertion-deletion setting. We also showed that only when combining arbitrary-length intervals with weights then a lower bound of $\Omega(n)$ can be obtained for constant factor approximations in the insertion-only setting.

We conclude with two open problems.

1. Our $(\frac{3}{2} + \epsilon)$-approximation insertion-only streaming algorithm and our $(2 + \epsilon)$-approximation insertion-deletion streaming algorithm for weighted unit-length intervals have space complexities that depend on $\frac{1}{\epsilon}$. Can we prove any lower bounds that capture these dependencies on $\frac{1}{\epsilon}$?
2. Interval Selection is relatively well understood by now. Can we obtain optimal algorithm for other geometric objects, such as axis-aligned unit squares or unit circles? For example, as previously mentioned, regarding axis-aligned unit squares, it is known that a 3-approximation can be obtained with space $O(|OPT|)$, while a lower bound shows that space $\Omega(n)$ is needed for approximation factors below 2.5 [5]. Can we close this gap?

References

1. Ainesh Bakshi, Nadiia Chepurko, and David P. Woodruff. Weighted maximum independent set of geometric objects in turnstile streams. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 64:1–64:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.64.
2. Sujoy Bhore, Fabian Klute, and Jelle J. Oostveen. On streaming algorithms for geometric independent set and clique. In Parinya Chalermsook and Bundit Laekhanukit, editors, *Approximation and Online Algorithms – 20th International Workshop, WAOA 2022, Potsdam, Germany, September 8-9, 2022, Proceedings*, volume 13538 of *Lecture Notes in Computer Science*, pages 211–224. Springer, 2022. doi:10.1007/978-3-031-18367-6_11.
3. Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. *Theoretical Computer Science*, 702:77–96, 2017. doi:10.1016/j.tcs.2017.08.015.
4. Graham Cormode, Jacques Dark, and Christian Konrad. Approximating the caro-wei bound for independent sets in graph streams. In Jon Lee, Giovanni Rinaldi, and Ali Ridha Mahjoub, editors, *Combinatorial Optimization – 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers*, volume 10856 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2018. doi:10.1007/978-3-319-96151-4_9.
5. Graham Cormode, Jacques Dark, and Christian Konrad. Independent Sets in Vertex-Arrival Streams. 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 45:1–45:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.45.
6. Graham Cormode and Donatella Firmani. A unifying framework for ℓ_0-sampling algorithms. *Distributed Parallel Databases*, 32(3):315–335, 2014. doi:10.1007/s10619-013-7131-9.
7. Yuval Emek, Magnus M. Halldorsson, and Adi Rosen. Space-Constrained Interval Selection. In *39th International Colloquium on Automata, Languages, and Programming (ICALP) 2012, Warwick, UK*, July 2012.
8. Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1363–1374. ACM, 2020. doi:10.1145/3357713.3384286.
9. Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming algorithms for independent sets in sparse hypergraphs. *Algorithmica*, 76(2):490–501, 2016. doi:10.1007/s00453-015-0051-5.
10. Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. Streaming and communication complexity of clique approximation. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 2012. doi:10.1007/978-3-642-31594-7_38.
11. Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636. IEEE Computer Society, 1996. doi:10.1109/SFCS.1996.548522.
12. Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *J. ACM*, 32(1):130–136, January 1985. doi:10.1145/2455.214106.

13 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '11, pages 49–58, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1989284.1989289.

14 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996. doi:10.1017/CBO9780511574948.

A Analysis of the Lower Bound Reduction From Section 4.2.1

As in the insertion-only case, we will first argue that the size of a largest independent set is very different depending on whether the answer bit in $\mathsf{AUG\text{-}CHAIN}_p(k)$ is 1 or 0.

▶ **Lemma 16.** *If the answer bit in the $\mathsf{AUG\text{-}CHAIN}_p(k)$ instance is 1 then the Interval Selection instance constructed in the reduction has an independent set of size p. If the answer bit is 0 then every independent set is of size at most 1.*

Proof. The properties we use here to determine the independence of intervals from different gadgets follow from Observation 13.

Let $\mathcal{K}_i$ denote the clique gadget introduced by party i after the deletions introduced by party $i+1$ have been applied.

Suppose first that the answer bit is 1. Then, the set of intervals $\{I_1, \ldots, I_p\}$, where I_i is the σ^ith interval of $\mathcal{K}_i$, is an independent set of size p.

Suppose now that the answer bit is 0. Then, it can be seen that all intervals in the instance intersect with the mid-point of the interval added by party p. The intervals thus form a clique, and, thus, every independent set is of size at most 1. ◀

Proof of Theorem 4. From Lemma 16, we see that if the algorithm $\mathcal{A}$ has an approximation factor $\alpha \leq p-1$ then party p can distinguish between the 0 and 1 case in $\mathsf{AUG\text{-}CHAIN}_p(k)$. Hence, we will set $p = \alpha + 1$.

By Theorem 9, algorithm $\mathcal{A}$ must use space $\Omega(\frac{k}{p^2}) = \Omega(\frac{k}{\alpha^2})$. Similar to the construction in Section 3.3, we have $\Delta = (4(k+1))^{p-1} = (4(k+1))^{\alpha}$, which implies $k = \frac{1}{4}\Delta^{\frac{1}{p-1}} - 1$ and gives a lower bound of $\Omega(\Delta^{\frac{1}{p-1}}/\alpha^2)$. Since the construction uses at most $n \leq k \cdot p = (\frac{1}{4}\Delta^{\frac{1}{p-1}} - 1)(\alpha+1)$ intervals, the lower bound cannot exceed $O(\frac{n}{\alpha^3})$.

Last, the construction is based on reals a, b. We pick any a and select b large enough such that the shortest interval is of length 1. This concludes the proof. ◀

Listing 4-Cycles

Amir Abboud
Weizmann Institute of Science, Rehovot, Israel

Seri Khoury
UC Berkeley, CA, USA

Oree Leibowitz
Weizmann Institute of Science, Rehovot, Israel

Ron Safier
Weizmann Institute of Science, Rehovot, Israel

Abstract

We study the fine-grained complexity of listing all 4-cycles in a graph on n nodes, m edges, and t such 4-cycles. The main result is an $\tilde{O}(\min(n^2, m^{4/3}) + t)$ upper bound, which is best-possible up to log factors unless the long-standing $O(\min(n^2, m^{4/3}))$ upper bound for *detecting* a 4-cycle can be broken. Moreover, it almost-matches recent 3-SUM-based lower bounds for the problem by Abboud, Bringmann, and Fischer (STOC 2023) and independently by Jin and Xu (STOC 2023). Notably, our result separates 4-cycle listing from the closely related *triangle* listing for which higher conditional lower bounds exist, and rule out such a "detection plus t" bound. We also show by simple arguments that our bound cannot be extended to mild generalizations of the problem such as reporting all pairs of nodes that participate in a 4-cycle.

Independent work: Jin and Xu [26] also present an algorithm with the same time bound.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Graph algorithms, cycles listing, subgraph detection, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.25

Funding *Amir Abboud*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No 101078482). Additionally, Amir Abboud is supported by an Alon scholarship and a research grant from the Center for New Scientists at the Weizmann Institute of Science.

1 Introduction

Finding small patterns in large graphs is a classical task. In the *detection* formulation we only want to decide if a small pattern H exists in a large graph G (as a not-necessarily induced subgraph); this is known as the *Subgraph Isomorphism* problem and is one of the most extensively studied problems in graph algorithms. Its time complexity has been investigated from various angles, including classical algorithms (e.g. [34, 22, 4, 28, 39]), parameterized complexity (e.g. [19, 32]), streaming algorithms (e.g. [24, 23]), distributed computing (e.g. [14, 13]), quantum algorithms (e.g. [31, 30]) and so on.[1]

In the *listing* formulation we are asked to return all occurrences of the pattern in the graph as efficiently as possible. This problem has received much attention both from the classic graph algorithms literature (e.g. [27, 8, 15, 9]) and also from the large literature on *enumerating answers to queries* in databases (where they focus on the closely related problem of listing all patterns with *small delay* between consecutive outputs), see [18, 37, 20, 10, 11, 12].

[1] An extensive list of citations for each of these topics is infeasible.

© Amir Abboud, Seri Khoury, Oree Leibowitz, and Ron Safier;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 25; pp. 25:1–25:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Our interest is in the *fine-grained complexity* of such problems and we therefore focus on the simplest patterns without near-linear time algorithms: triangles and 4-cycles. Besides the theoretical interest, the enumeration of short cycles has received much attention due to applications, see e.g. [6, 7, 21, 38].

Throughout the paper, we assume that graphs are unweighted, undirected, and have n nodes and m edges unless stated otherwise. The classical *Johnson's Algorithm* [27] lists all cycles in a graph in time that is polynomial in n and in their number t. Four decades later, its time complexity was improved significantly so that it runs in the optimal $O(m+t)$ time [8]. One may hope that such an efficient algorithm can list all k-cycles for a *fixed* k such as $k=3$ or $k=4$ in time $O(m+t)$ (here t is the number of k-cycles). Unfortunately, this is not known and is impossible under popular conjectures as we discuss below.

A simple observation (discussed in Section 3.1) shows that we cannot hope for an $O(T(n,m)+t)$ upper bound for listing unless we can solve the detection problem in time $O(T(n,m))$. Moreover, note that the $+t$ term is necessary because typically the number of occurrences of a pattern could be much larger than $T(n,m)$. Thus, our starting point is the state-of-the-art for *detecting* triangles and 4-cycles. The best known algorithms for *detecting* if a graph has at least one pattern take $O(\min(n^\omega, m^{2\omega/(\omega+1)}))$ for triangle [4], where $\omega < 2.37188$ is the fast matrix multiplication exponent [3, 17], and $O(\min(n^2, m^{4/3}))$ for 4-cycle [4]. Note that if $\omega = 2$ then the two bounds are the same. Thus, the main question is whether we can solve the listing problem with a time complexity that matches these bounds, plus t. In particular, can we list all 4-cycles in $O(\min(n^2, m^{4/3})+t)$ time?

Such a result is unlikely for triangle listing. A reduction by Kopelowitz, Pettie, and Porat [29] that optimizes a construction by Pătraşcu [35] shows that an $O(n^\omega + t)$ bound or any $O(min(n^{3-\varepsilon}, m^{3/2-\varepsilon})+t)$ algorithm for any $\varepsilon > 0$ would refute the 3-SUM Conjecture[2].

The actual state-of-the-art for triangle listing is more complicated. An exhaustive search lists all triangles in $O(n^3)$ time. An algorithm of time $O(m^{\frac{3}{2}})$ can be achieved by a simple application of low-degree/high-degree separation technique (see for example [4]). A work by Chiba and Nishizeki [15] shows how to list all triangles (and all 4-cycles) in $O(ma(G))$ time where $a(G)$ is the arboricity[3] of the graph G. This running time is essentially the same as $O(m^{\frac{3}{2}})$ because $a(G) \leq m^{1/2}$, but it shows an improvement for graphs with low arboricity (such as planar graphs). These algorithms are essentially optimal because a graph may contain $\Theta(\min(n^3, m^{\frac{3}{2}}))$ triangles, but better results can be achieved if we also take the number of triangles into account. Then, denoting the number of triangles in the graph by t, a more clever algorithm by Björklund, Pagh, Vassilevska William, and Zwick [9] solves triangle listing in time $\tilde{O}(n^\omega + n^{\frac{3(\omega-1)}{5-\omega}} t^{\frac{2(3-\omega)}{5-\omega}})$ and $\tilde{O}(m^{\frac{2\omega}{\omega+1}} + m^{\frac{3(\omega-1)}{\omega+1}} t^{\frac{3-\omega}{\omega+1}})$. Assuming $\omega = 2$ the running times simplify to $\tilde{O}(n^2 + nt^{2/3})$ and $\tilde{O}(m^{4/3} + mt^{1/3})$. A matching tight (assuming $\omega = 2$) lower-bound under the 3-SUM Conjecture exists [35, 29] and also, more recently, under the APSP Conjecture[4] [41].

The hardness of achieving "detection plus t" time for triangle listing may suggest that we cannot achieve the desired $O(\min(n^2, m^{4/3})+t)$ time for 4-cycle listing. Indeed, in Section 3 we present some simple observations showing that such a bound is conditionally impossible

[2] The 3-SUM problem is as follows: Given a set $A \subseteq \{-n^3, \ldots, n^3\}$ of n elements, decide if there are three distinct elements $a, b, c \in A$ such that $a+b+c=0$. The 3-SUM Conjecture is the conjecture that solving the 3-SUM problem requires $n^{2-o(1)}$ time.

[3] The arboricity of a graph is the minimum number of edge-disjoint spanning forests into which the graph can be decomposed.

[4] All-Pairs Shortest Paths (APSP): Given an n-node edge-weighted directed graph with positive weights in $\{1, \ldots, n^{o(1)}\}$, compute for all pairs of nodes the shortest distance between them. The APSP Conjecture is the conjecture that solving All-Pairs Shortest Paths requires $n^{3-o(1)}$ time.

for mild generalizations of the problem such as in directed graphs, or if the task is to count the number of 4-cycles rather than list them (without a $+t$ term), or if our task is to report which pairs of nodes participate in 4-cycles. Fortunately, none of these reductions give a lower bound in the basic setting of the problem.

Our main result is a positive resolution to the above question, achieving the best-possible "detection plus t" bound for 4-cycle listing. Before our work, only the trivial $O(n^3+t)$ bound was known.

▶ **Theorem 1.** *4-cycle listing can be solved in* $O(\min(n^2+t,(m^{4/3}+t)\cdot\log^2 n))$ *time.*

Notably, this running time is faster than the aforementioned lower bound for triangle listing stating that there is no $O(min(n^{3-\varepsilon}, m^{3/2-\varepsilon})+t)$ algorithm for triangle listing for any $\varepsilon > 0$ (assuming 3-SUM) and therefore separates the two problems.

Any improvement on our upper bound would break the longstanding upper bound for 4-cycle *detection*, as discussed in Section 3.1. Moreover, a recent "short cycle removal" technique in fine-grained complexity introduced by Abboud, Bringmann, Khoury, and Zamir [2], can prove conditional lower bounds for 4-cycle listing that holds even if 4-cycle detection turns out to be easier. In particular, two independent papers [1, 26] use this technique to prove a $(\min(n^2, m^{4/3})+t)^{1-o(1)}$ lower bound under the 3-SUM Conjecture, showing that our algorithm is tight.[5] One interesting aspect of our result (mentioned in [1]) is that it proves, in some sense, the optimality of the lower-bound machinery in these recent works.

The main technical ingredient underlying our results, which distinguish 4-cycle from triangle and from the variants discussed in Section 3 is a (well-known) *supersaturation* result stating that if the number of edges in a graph is larger than a certain threshold, then the graph must have a large number of 4-cycles.

Independent work: Jin and Xu [26] independently obtained an algorithm for 4-cycle listing of the same running time (they have fewer log factors) by a similar approach, but some lemmas are proved differently.[6]

1.1 Outline of the Paper

We organize this paper as follows. In Section 2 we present and analyze an efficient algorithm for 4-cycle listing. In Section 3 we discuss some hardness results for mild generalizations of the problem. Lastly, in Section 4 we discuss two related open problems which we find interesting. The first open problem is regarding the generalization of our results for longer even cycles. The second open problem is determining which detection problem is harder: triangle or 4-cycle detection.

2 Upper Bounds for 4-Cycle Listing

In Section 2.1, we start with a simple extension of the folklore $O(n^2)$-time algorithm for 4-cycle detection [36] to an $O(n^2+t)$-time algorithm for 4-cycles listing, where t is their number.

[5] These lower bounds are for *enumeration* with constant *delay*; our upper bounds also apply in that formulation of the listing problem by standard techniques. We choose to present our results for listing rather than enumeration (which would have made the theorems slightly stronger) because it makes our analysis more intuitive.

[6] The two papers appeared simultaneously on ArXiv, but their results were included in a paper with more results that were published in STOC 2023; their main results were lower bounds.

Then, in Section 2.2, we present our main result, which is an $\tilde{O}(m^{4/3} + t)$-time algorithm for 4-cycles listing.

2.1 Warm-up: An $O(n^2 + t)$-time Algorithm

The simple $O(n^2)$-time detection algorithm can be described as follows. We enumerate all 2-paths (u, x, v) in the graph until we notice that some pair (u, v) has appeared as the endpoints of more than a single 2-path; when that happens, we stop and report the existence of a 4-cycle (since two 2-paths sharing the same endpoints form a 4-cycle). While enumerating the 2-paths, the algorithm marks in an $n \times n$ table all the pairs (u, v) that have already been seen as endpoints of 2-paths, and therefore it knows when to stop. Importantly, the algorithm never enumerates more than $n^2 + 1$ 2-paths before stopping, giving the desired upper bound on the running time.

Another important point implicit in the above description is that all 2-paths can be enumerated efficiently, in linear time in m and the number of 2-paths listed so far: For each node a, for each neighbor $b \in N(a)$, for each neighbor $c \in N(b)$ enumerate the 2-path (a, b, c).

The $O(n^2 + t)$ listing algorithm does exactly the same thing, except it does not stop after finding the first 4-cycle but continues enumerating all the 2-paths of the graph. (Additionally, it records all 2-paths in the entry of the table corresponding to a pair of nodes (u, v) so that all 4-cycles involving (u, v) can be listed.) The following observations analyze the running time.

We start with a worst-case bound on the number of 2-paths in a graph, as a function of the number of 4-cycles.

▶ **Observation 2.** *Given a graph G, the number of 2-paths in G is at most $n^2 + 4t$ where t is the number of 4-cycles in G.*

Proof. Let p be the total number of 2-paths in the graph and, for a pair of nodes u, v, let $p_{u,v}$ be the number of 2-paths between u and v and notice that

$$4t = 2 \sum_{u,v \in V} \binom{p_{u,v}}{2} \geq \sum_{\substack{u,v \in V \\ p_{u,v} > 1}} p_{u,v} = p - \sum_{\substack{u,v \in V \\ p_{u,v} = 1}} p_{u,v} \geq p - \binom{n}{2} \geq p - n^2. \qquad ◀$$

This leads to the $O(n^2 + t)$ bound on the simple algorithm above that lists all 2-paths in the graph, as formalized in the following observation.

▶ **Observation 3.** *Given a graph G, there is an $O(n^2 + t)$-time algorithm that lists all the 4-cycles, where t is their number.*

Proof. Recall the algorithm above that lists all 2-paths (u, x, v) and indexes them by their endpoints (u, v), and then lists all 4-cycles formed by two 2-paths sharing the same pair (u, v) as endpoints. First, note that the algorithm lists each 4-cycle (u, x, v, y) exactly twice (once for the pair (u, v) and once for the pair (x, y)). The time complexity is thus upper bounded by a constant factor times the number of 2-paths, plus the number of 4-cycles t, plus the number of pairs n^2. By the upper bound on the number of 2-paths from Observation 2, we get the $O(n^2 + t)$ bound. ◀

2.2 An $\tilde{O}(m^{4/3} + t)$-time Algorithm

In order to improve the $O(n^2 + t)$-time algorithm, we can't afford listing all 2-paths. For instance, in a star graph, there are $O(n^2)$ 2-paths, but no 4-cycles. Hence, intuitively speaking, we need to narrow our attention to a certain type of 2-paths that are useful for

listing 4-cycles efficiently. Indeed, to overcome the star example, it suffices to note that there is no point in listing 2-paths with endpoints of degree one (leaves), as these 2-paths can't be extended to 4-cycles.

To extend this intuition, perhaps one could try to split the nodes into low-degree and high-degree groups, denoted by L and H, respectively, and consider different types of 2-paths with respect to this partitioning. The advantage of such partitioning is that we can narrow our attention to specific types of 2-paths that are more challenging for listing (i.e., types of 2-paths that could be too expensive to list). For instance, one can immediately spot two types of 2-paths that can be listed efficiently: 2-paths with a low-degree node at the center, and 2-paths that use only high-degree nodes. Indeed, to list all the 2-paths with a low-degree node at the center, we can go over all the edges e incident to a low-degree node, and list the corresponding 2-paths with a low-degree node at the center that e is part of. This takes $O(m\Delta)$ time, where Δ is the degree threshold. To list all the 2-paths that use only high-degree nodes H, we can use the $O(|H|^2 + t) = O((m/\Delta)^2 + t)$-time algorithm from Observation 3. By picking $\Delta = m^{1/3}$, listing these two types of 2-paths takes $O(m^{4/3} + t)$ time (as shown in Lemma 8). Furthermore, listing these two types of 2-paths suffices for listing all types of 4-cycles, except the 4-cycles that use two overlapping 2-paths of the form LHH (2-paths with a high degree node at the center, one low degree endpoint, and one high degree endpoint). That is, these 4-cycles are of the form $LHHL$. To list these 4-cycles, we need to find a way to list 2-paths of the form LHH.

Unfortunately, we can't afford listing all 2-paths of the form LHH. For instance, take a graph where there is a node u that is connected to $\Theta(n)$ leaves (low degree nodes) and to $\Theta(n^{2/3-\varepsilon})$ high-degree nodes, where each of these high-degree nodes is connected to $\Theta(n^{1/3+\varepsilon})$ leaves. In this example, we have $m = O(n)$ edges, $m^{5/3-\varepsilon} \gg m^{4/3}$ 2-paths of the form LHH (the ones that go through u), but no 4-cycles.

To overcome such examples, recall that the only remaining type of 4-cycles that we need to list are the ones of the form $LHHL$. Since we know how to list 2-paths of the form HLL (2-paths with a low-degree node at the center) efficiently, it suffices to list *only one of the two* LHH 2-paths that such a 4-cycle consists of. Hence, for the 4-cycles of the form $LHHL$, one could wonder: is there a property that one of the two overlapping LHH paths (that the 4-cycle consists of) must have, that would make it easier to list such 4-cycles?

Indeed, given a 4-cycle of the form $LHHL$, consider the two middle high-degree nodes. One of them must have a degree greater or equal to the other. Therefore, it suffices to list 2-paths of the form LHH, where the degree of the middle node is at most the degree of the third node (the high-degree endpoint). We refer to such 2-paths as $L \to H \to H$ (the orientation from a node u to a node v here means that u's degree is at most v's). The question that remains is: can we afford to list all 2-paths of the form $L \to H \to H$? Interestingly, we answer this question affirmatively. Roughly speaking, we show that there can't be too many $L \to H \to H$ paths compared to 4-cycles, unless the number of such 2-paths was small enough for our purposes. Hence, we use a charging argument that allows us to list all such 2-paths.

For instance, going back to the graph construction we discussed with $m^{5/3-\varepsilon}$ LHH 2-paths and no 4-cycles, it contains only $O(m)$ $L \to H \to H$ 2-paths.

A road-map for the technical parts. First, in Section 2.3, we prove a helpful theorem that shows that there can't be too many 2-paths of the form $L \to H \to H$ compared to 4-cycles. We refer to this theorem as the $L \to H \to H$ theorem. Then, in Section 2.4, we put everything together and prove our main result - an $\tilde{O}(m^{4/3} + t)$-time algorithm for 4-cycle listing.

2.3 The $L \to H \to H$ Theorem

In this section, we prove the following theorem that connects the number of 4-cycles in a graph to the number of 2-paths of a certain type. The degree of a node v is denoted by $\deg(v)$.

▶ **Theorem 4.** *Given an undirected graph $G = (V, E)$ with m edges, let H be the set of nodes with degree larger than $m^{1/3}$, and $L = V \setminus H$. Orient all the edges $\{u, v\} \in E$ from u to v if $\deg(u) \leq \deg(v)$ (break ties arbitrarily). Let P be the number of directed 2-paths of the form $L \to H \to H$ in G. It holds that if $P > 100m^{4/3} \log^2 m$, then there are at least $P/(100 \log^2 m)$ 4-cycles in G.*

In order to prove Theorem 4, we use two helper lemmas. In Lemma 5, we show that the number of 4-cycles is $\Omega(d^4 - n^2)$, where d is the average degree. In Lemma 6, we provide a view of the graph that has some nice properties. In particular, this view is a partitioning of the graph that has a useful regularity property, while the number of $L \to H \to H$ 2-paths is preserved. The proof of Theorem 4 is provided after the proof of Lemma 6.

▶ **Lemma 5.** *Any graph with n nodes and average degree d has $\Omega(d^4 - n^2)$ 4-cycles.*

Proof. Let $G = (V, E)$ be a graph with n nodes and average degree d. Let A be the adjacency matrix of G. Denote by $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ the n eigenvalues of A. The top eigenvalue λ_1 of A is at least d. This is because:

$$\lambda_1 = \max_{v^T v = 1} v^T A v$$

Now, consider $u = (1/\sqrt{n}, \ldots, 1/\sqrt{n})$. Clearly, it holds that $u^T u = 1$. On the other hand, we have that:

$$u^T A u = \sum_{w \in V} \frac{1}{n} \deg(w) = 2|E(G)|/n = d$$

The number of closed 4-walks in G is at least d^4. This is because this number is exactly the trace of A^4 and

$$\mathrm{tr}(A^4) = \sum_{i=1}^{n} \lambda_i^4 \geq \lambda_1^4 \geq d^4$$

Let T be the number of 4-cycles. To finish the proof, we show that the number of closed 4-walks is at most

$$3T + 2n^2 \tag{1}$$

This would imply that $T \geq (d^4 - 2n^2)/3$ which would imply that $T = \Omega(d^4 - n^2)$. To prove Equation 1, notice that the distinct types of closed 4-walks are exactly 4-cycles, 2-paths and edges. There are T 4-cycles and at most n^2 edges. By Observation 2, the number of 2-paths is bounded by $n^2 + 2T$. ◀

▶ **Lemma 6.** *Given a graph $G = (V, E)$ with m edges, let H be the set of nodes with degree larger than $m^{1/3}$, and $L = V \setminus H$. Orient all the edges $\{u, v\}$ from u to v if $\deg(u) \leq \deg(v)$ (break ties arbitrarily). Let P be the number of directed 2-paths of the form $L \to H \to H$. There is a partition of a subset of H into two parts H_1 and H_2 (i.e. $H_1 \cap H_2 = \emptyset$ and $H_1 \cup H_2 \subseteq H$), such that:*

1. *The number of directed 2-paths of the form $L \to H_1 \to H_2$ is at least $P/(4\log^2 n)$.*
2. *Each node in H_1 has the same number of incoming edges from L up to a multiplicative 2-factor.*
3. *Each node in H_1 has the same number of outgoing edges to H_2 up to a multiplicative 2-factor.*
4. *Each node in H_2 has at least one incoming edge from H_1.*

Proof. We start by describing a simpler partitioning. This simpler partitioning splits the set of high-degree nodes H into two sets H_1' and H_2' such that the number of 2-paths of the form $L \to H_1' \to H_2'$ is at least $P/4$. Such a partitioning exists by the probabilistic method: Each node in H joins H_1' with probability 1/2 independently and otherwise it joins H_2'. Thus, the probability that a 2-path $u \to v \to w$ of the form $L \to H \to H$ survives in $L \to H_1' \to H_2'$ is $\Pr(u \in H_1' \wedge v \in H_2') = 1/4$. Hence, in expectation, we get $P/4$ such paths in $L \to H_1' \to H_2'$.

Next, for any node $u \in H_1'$, let $\deg_{left}(u)$ be the number of incoming edges from L. Similarly, let $\deg_{right}(u)$ be the number of outgoing edges to H_2'. Let $B_{ij} = \{u \in H_1' : \deg_{left}(u) \in [2^i, 2^{i+1}] \wedge \deg_{right}(u) \in [2^j, 2^{j+1}]\}$. It holds that the number of 2-paths of the form $L \to H_1' \to H_2'$ is

$$P/4 = \sum_{u \in H_1'} \deg_{left}(u) \cdot \deg_{right}(u) = \sum_{i,j \in [\log n]} \sum_{u \in B_{ij}} \deg_{left}(u) \cdot \deg_{right}(u)$$

Hence, there are $i', j' \in [\log n]$ such that

$$\sum_{u \in B_{i'j'}} \deg_{left}(u) \cdot \deg_{right}(u) \geq P/(4\log^2 n),$$

which implies that there are at least $P/(4\log^2 n)$ $L \to H_1' \to H_2'$ 2-paths that go through $B_{i'j'}$. Let $H_1 := B_{i'j'}$ and define H_2 to be all the vertices in H_2' with at least one incoming edge from H_1. The sets L, H_1, and H_2 satisfy the desired properties. ◀

Now we are ready to prove Theorem 4.

Proof of Theorem 4. First, take the partitioning from Lemma 6. We know that the number of 2-paths of the form $L \to H_1 \to H_2$ is $P' \geq P/(4\log^2 m) \geq 25m^{4/3}$. Recall that each node in H_1 has the same in-degree from L up to a multiplicative 2-factor. Denote by d_{left} the minimum over these degrees. Similarly, each node in H_1 has the same out-degree to H_2 (up to a multiplicative 2-factor). Denote the minimum over these degrees by d_{right}. Furthermore, each node in H_2 has at least one incoming neighbor from H_1.

We show that the number of 4-cycles is $\Omega(P')$. For this, let d_0 be the average degree of the subgraph of G induced by the nodes in $H_1 \cup H_2$. Since we have at most $2m^{2/3}$ nodes in $H_1 \cup H_2$, it suffices to show, by Lemma 5, that $d_0^4 = \Omega(P')$.

For this, we split the proof into two cases:

Case 1: $|H_1| > |H_2|$. Observe that in this case, $d_0 > d_{right}/2$. Hence, it suffices to show that $d_{right}^4 = \Omega(P')$. For this, recall that $P' \leq 4|H_1| \cdot d_{left} \cdot d_{right}$ (where the 4 factor is coming from the two 2 factors for d_{left} and d_{right}), and assume towards a contradiction that $(d_{right})^4 < P'$, which implies (by substituting d_{right} with $P'/(4|H_1| \cdot d_{left})$) that $(P')^3 < (4|H_1| \cdot d_{left})^4$. Therefore, we get that $P' < 16m^{4/3}$ (as $|H_1| \cdot d_{left} \leq m$), which is a contradiction.

Case 2: $|H_1| \leq |H_2|$. In this case, we have that $d_0 > d_{right} \cdot |H_1|/(2|H_2|)$. Hence, it suffices to show that $(d_{right}|H_1|/2|H_2|)^4 > P'$. Assume towards a contradiction that $(d_{right}|H_1|/2|H_2|)^4 \leq P'$. By substituting d_{right} with $P'/(4d_{left} \cdot |H_1|)$, this implies that $(P')^3 < (8d_{left} \cdot |H_2|)^4$. Now we want to argue that $d_{left} \cdot |H_2|$ is at most m to get a contradiction to $P' \geq 25m^{4/3}$. For this, recall that each node in H_2 has at least one incoming edge from H_1, which implies that the degree (in G) of each node in H_2 is at least d_{left}. Hence, we have that $m \geq \sum_{u \in H_2} \deg(u) \geq \sum_{u \in H_2} d_{left} = |H_2| \cdot d_{left}$, as desired. ◀

2.4 Listing 4-Cycles

In this section, we prove the following theorem.

▶ **Theorem 7.** *Listing all the 4-cycles in an undirected graph $G = (V, E)$ can be done in $O(m^{4/3} \log^2 m + t \log^2 m)$ time, where m is the number of edges and t is the number of 4-cycles.*

The proof of Theorem 7 is based on listing several types of 2-paths efficiently. Lemma 8 shows that we can list each of these types of 2-paths efficiently.

▶ **Lemma 8.** *Given a graph $G = (V, E)$ with m edges and t 4-cycles. Let H be the set of nodes with degrees larger than $m^{1/3}$ and $L = V \setminus H$.*

1. *Listing all the 2-paths with nodes only from H can be done in $O(m^{4/3} + t)$ time.*
2. *Listing all the 2-paths with a node from L at the center can be done in $O(m^{4/3})$ time.*
3. *Orient all the edges $\{u, v\}$ from u to v if $\deg(u) \leq \deg(v)$ (break ties arbitrarily). Listing all the directed 2-paths of the form $L \to H \to H$ can be done in $O(m^{4/3} \log^2 m + t \log^2 m)$ time.*

Proof.

1. Let G' be the subgraph of G induced by H. Denote by n' the number of nodes in G' and by t' the number of 4-cycles in G'. Observe that $n' \leq 2m^{2/3}$ and $t' \leq t$. By using an argument similar to the one used in Observation 3, we can list all the 2-paths in G' in time $O(n'^2 + t') = O(m^{\frac{4}{3}} + t)$.
2. Scan all the edges in G and for those with at least one endpoint in L scan all the neighbors of the endpoints in L. This procedure finds all the 2-paths with a node from L in the middle. Each node in L has at most $m^{1/3}$ neighbors. Therefore, the running time of this procedure is $O(m^{\frac{4}{3}})$.
3. We can list all the $L \to H \to H$ 2-paths in time that is linear in their number and the number of edges. This can be done by going over all the nodes $u \in L$, and then going over the neighbors $v \in H$ of u, and then going over all neighbors of v with higher degree than v. This can be done in time that is linear in the number of edges and the number of $L \to H \to H$ 2-paths. This is because we can prepare a set of higher degree nodes in H for all the nodes $u \in V$, via a simple $O(m)$-time preprocessing step, where we go over all the edges (with at least one endpoint in H), detect for each edge the higher degree endpoint, and store it. Therefore, the running time of the algorithm is $O(m + P)$ where P is the number of directed $L \to H \to H$ 2-paths. Since by Theorem 4 we have that $P = O(m^{4/3} \log^2 n + t \log^2 n)$, the claim follows. ◀

Proof of Theorem 7. We consider all the different types of 4-cycles (in terms of low-high degree nodes) and show that we can list all of them in the desired running time.

Type 1: 4-cycles that use only high-degree nodes. This class of 4-cycles can be decomposed into two 2-paths of all high-degree nodes. These 2-paths can be listed in $O(m^{4/3}+t)$ time as shown in Lemma 8.

Type 2: 4-cycles with two non-adjacent low-degree nodes. This class of 4-cycles can be decomposed into two 2-paths with a low-degree node at the center. Theses 2-paths can be listed in $O(m^{4/3})$ time as shown in Lemma 8.

Type 3: 4-cycles with three high-degree nodes and one low-degree node. This class can be decomposed into a 2-path of all high-degree nodes and a 2-path with a low-degree node at the center. Using Lemma 8 these 2-paths can be listed in $O(m^{4/3}+t)$ time.

Type 4: 4-cycles with two adjacent low-degree nodes and two adjacent high-degree nodes. These 4-cycles can be decomposed into a directed $L \to H \to H$ 2-path and an LLH 2-path. Using Lemma 8 we can list all these 2-paths in $O(m^{4/3}\log^2 n + t \cdot \log^2 n)$ time.

To sum up, we showed how to list all the different types of 4-cycles in $O(m^{4/3}\log^2 n + t \cdot \log^2 n)$ time, as desired. ◀

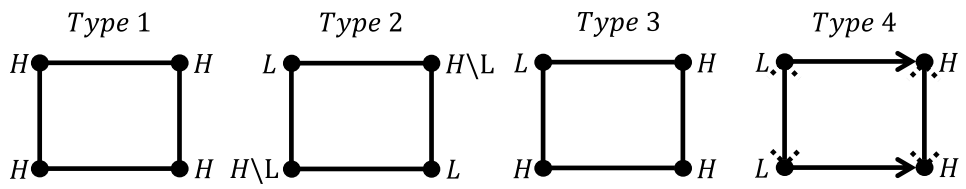


Figure 1 The types of 4-cycles. A 4-cycle of the fourth type always consists of an $L \to H \to H$ directed 2-path and a 2-path with a low-degree node at the center.

3 Hardness Results

In this section, we give a set of hardness results that more or less follow from folklore reductions (e.g. similar to the hardness reductions for general cyclic queries in databases [5]).

3.1 Lower Bound for 4-Cycle Listing from 4-Cycle Detection

In this section we give a general argument showing that listing cannot be solved faster than the "detection plus t" bound. We will prove it specifically for 4-cycle but the proof is more general.

▶ **Observation 9.** *If 4-cycle listing can be solved in time $O(T(n,m)+t)$ when t is the number of 4-cycles in the graph, then 4-cycle detection can be solved in time $O(T(n,m))$.*

Proof. We will show a fine-grained reduction from 4-cycle detection to 4-cycle listing. Given a graph G, run the 4-cycle listing algorithm on G. If the algorithm terminates in $O(T)$ time then if the algorithm outputs at least one 4-cycle then we also detected a 4-cycle and otherwise, there isn't a 4-cycle in G. If the algorithm takes more than cT time for large enough constant c, then we know that there are at least T 4-cycles in G. Therefore we detected a 4-cycle. In conclusion, we have an $O(T)$ time algorithm for 4-cycle detection as we wanted. ◀

Note that even an $o(\min(n^2, m^{4/3}))$ time algorithm for 4-cycle is not known [42, 16], so our listing upper bound is tight up to the polylog factor.

3.2 Hardness of All-Pairs 4-Cycle

In this section, we define a new natural problem which we call *all-pairs 4-cycle*. We show an algorithm for this problem and a conditional lower-bound from triangle-finding.

▶ **Definition 10** (all-pairs 4-cycle). *Given an undirected graph G, all-pairs 4-cycle is the problem of deciding for each pair of distinct vertices $u, v \in V(G)$ whether there is a 4-cycle (u, x, v, y) in G (for $x, y \in V(G)$).*

Note that if 4-cycle listing is in time $O(T)$ then all-pairs 4-cycle is also in time $O(T)$, as there is a trivial reduction from all-pairs 4-cycle to 4-cycle listing. But, we will see that this problem cannot be solved in $n^{\omega-\varepsilon}$ time as long as triangle detection cannot be solved in $n^{\omega-\varepsilon}$ time.

▷ Claim 11. There is an algorithm for all-pairs 4-cycle in $O(n^\omega)$-time. In addition, if there is an $O(n^{2+\varepsilon})$ time algorithm for all-pairs 4-cycle then there is an $O(n^{2+\varepsilon})$ time algorithm for triangle detection.

Proof. The all-pairs 4-cycle problem can be solved in $O(n^\omega)$ time by simply computing A^2 where A is the adjacency matrix of G and checking for any pair u, v whether $A^2[u, v] \geq 2$.

Assume there is an $O(n^{2+\delta})$ time algorithm (for $\delta \geq 0$) for all-pairs 4-cycle. We will show that we can solve triangle detection in $O(n^{2+\delta})$ time. Let G be an n-node graph. We define G^* as an extension of G by adding a new vertex x and connecting it to all the vertices in G. If there is a 4-cycle (u, t, v, w) in G^*, and $uv \in E(G)$ then u, v participate in a triangle in G (because at most one of the vertices t, w can be x). For the other direction, if u, v participate in a triangle (u, t, v) then after adding x they must participate in a 4-cycle (u, t, v, x) and $uv \in E(G)$. Therefore, we can solve triangle detection by running all-pairs 4-cycle on G^*, and check if there is a pair of distinct vertices $u, v \in V(G)$ such that all-pairs 4-cycle returned "yes" on them and $u, v \in E(G)$. The time required is $O(n^{2+\delta} + n^2) = O(n^{2+\delta})$. ◁

3.3 4-Cycle Counting is Triangle Counting Hard

We now present a folklore reduction from triangle counting to 4-cycle counting.

▶ **Observation 12.** *There is a fine-grained reduction from triangle counting to 4-cycle counting. The reduction time is quadratic in n and linear in $n + m$.*

Proof. We will show a reduction from triangle counting to 4-cycle counting. Let $G = (V, E)$ be a graph and our goal is to count the triangles in G. We construct a new graph G' by duplicating the vertices of G three times into three parts, and connecting by an edge a pair of vertices from distinct parts if they are a duplication of adjacent vertices in G. More formally, $G' = (V_A \cup V_B \cup V_C, E')$ where $V_A = \{v_A : v \in V\}$, $V_B = \{v_B : v \in V\}$, $V_C = \{v_C : v \in V\}$ and $E' = \{u_A v_B : uv \in E\} \cup \{u_B v_C : uv \in E\} \cup \{u_C v_A : uv \in E\}$. Clearly, every triangle (u, v, w) in G has exactly six matching triangles in G'.

Let G'' be the graph formed from G' by duplicating V_A and put edges between a vertex in V_A to its duplication. That is, $G'' = (V_A \cup V_{A'} \cup V_B \cup V_C, E'')$ such that $V_{A'} = \{v_{A'} : v_A \in V_A\}$ is a duplication of v_A, and the set of edges E'' is defined as follows: $E'' = \{v_A v_{A'} : v_A \in A\} \cup (E' \cap V_A \times V_C) \cup (E' \cap V_B \times V_C) \cup \{u_{A'} v_B : u_A v_B \in E'\}$. Notice that any triangle in G' becomes a 4-cycle in G''. But, there may be 4-cycles in G'' that are not corresponding to triangles in G'. However, these 4-cycles can't include edges in $\{v_A v_{A'} : v_A \in A\}$. Let G^* be G'' after removing the edges in $\{v_A v_{A'} : v_A \in A\}$. The 4-cycles in G^* are exactly the

4-cycles in G'' that are not corresponding to triangles in G'. So, by counting the number of 4-cycles in G'' and subtracting the number of 4-cycles in G^* we get the exact number of triangles in G'. Since there are six triangles in G' for each triangle in G, we can get the number of triangles in G.

The running time of the reduction is $O(n+m) = O(n^2)$ for constructing the graphs G', G'' and G^*. ◀

▶ **Corollary 13.** *An $O(\min(n^{\omega-\varepsilon}, m^{\frac{2\omega}{\omega+1}-\varepsilon}))$-time algorithm for 4-cycle counting (for some $\varepsilon > 0$) would imply an $O(\min(n^{\omega-\varepsilon}, m^{\frac{2\omega}{\omega+1}-\varepsilon}))$-time algorithm for triangle counting.*

3.4 Hardness of Directed 4-Cycle Listing from Triangle Listing

With a similar reduction to the previous one, we can show the hardness of 4-cycle listing in *directed* graphs via a reduction from triangle listing.

▶ **Definition 14** (directed 4-cycle listing)**.** *Given a directed graph G, directed 4-cycle listing is the problem of listing all the directed 4-cycles in G.*

▷ Claim 15. If directed 4-cycle listing is in time $O(T(n,m)+t)$ then triangle listing is also in time $O(T(n,m)+t)$.

Proof. Given a triangle listing instance G we create a graph G'' as in the previous proof. Now, we create a *directed* graph G''' from G'' by directing all the edges as follows: $V_A \to V_{A'} \to V_B \to V_C \to V_A$. Thus, any directed 4-cycle in this new graph corresponds to a triangle in G', and recall that any triangle in G becomes six triangles in G'. Therefore, the number of 4-cycles in G''' and the number of triangles in G have a one-to-six correspondence, and so any directed 4-cycle listing algorithm can be used to list triangles with an additional factor six overhead in t. ◁

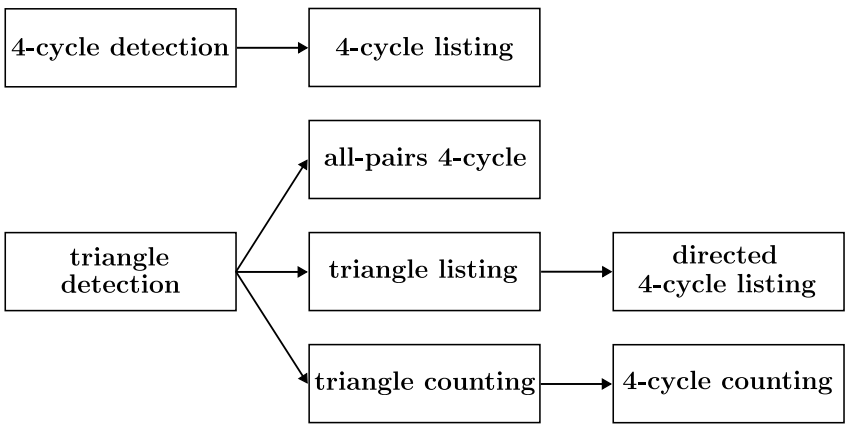


Figure 2 Overview of the reductions in Section 3.

4 Open Problems

We conclude with two open problems that we find interesting.

4.1 Generalization for Larger Even Cycles

A natural follow-up question for our work is, how hard is it to list all the cycles of length $2k$ for constant $k > 2$?

Yuster and Zwick developed an algorithm for $2k$-cycle detection in time $O(n^2)$ [42]. Later on, Dahlgaard et al. showed an algorithm in time $O(m^{\frac{2k}{k+1}})$ [16]. It is natural to ask whether can we generalize our 4-cycle listing algorithm to an algorithm for $2k$-cycle listing for any $k \geq 2$ in the same time as $2k$-cycle detection plus the number of cycles.

▶ **Open Question 16.** *Can $2k$-cycle listing be solved in time $O(n^2 + t)$ where n is the number of vertices and t is the number of $2k$-cycles in G?*

▶ **Open Question 17.** *Can $2k$-cycle listing be solved in time $O(m^{\frac{2k}{k+1}} + t)$ where m is the number of edges and t is the number of $2k$-cycles in G?*

The $O(n^2 + t)$-algorithm for 4-cycle listing is pretty simple: just list all the 2-paths and construct 4-cycles from the 2-paths. The key observation of the analysis is that any new 2-path we found after n^2 2-paths were already found must form at least one more 4-cycle.

A naive candidate for an algorithm for $2k$-cycle listing in $O(n^2 + t)$ is to list all the k-paths and construct all the $2k$-cycles from them by looking for pairs of k-paths between a certain pair of vertices. But notice that there is a problem. In the 4-cycle case, any pair of 2-paths between a certain pair of vertices form a 4-cycle. However, this is not the case in the general $2k$-cycle case. For example, a pair of 3-paths between a certain pair of vertices may collide in the middle and thus form a 4-cycle with an additional edge instead of a 6-cycle. Therefore, there can be too many 3-paths but not enough 6-cycles.

For regular graphs, this simple algorithm achieves $O(n^2 + t)$ time. The analysis is simple: The number of k-paths is at most nd^k. If $d = O(n^{1/k})$ this number is at most $O(n^2)$. Otherwise, the graph is "supersaturated" and the number of $2k$-cycles is $t = \Omega(d^{2k})$ [25, 33] and $t > nd^k$ (otherwise $d = O(n^{1/k})$). Therefore, in this case, the number of k-paths is at most $O(t)$. Hence, we can always bound the number of k-paths by $O(n^2 + t)$ and bound the running time of the algorithm by $O(n^2 + t)$.

To show that this simple algorithm doesn't work for general graphs we show the following example. Consider the following 4-layered graph $G = (A \cup B \cup \{u\} \cup D, E)$, such that $A = B = D = \Theta(n)$, which is defined as follows. $A \cup B$ is a bipartite regular graph with degree $n^{1/3-\varepsilon}$ and $n^{2-6\varepsilon}$ 6-cycles. This graph exists by a probabilistic argument. Consider a random bipartite graph such that each edge exists with probability $n^{-2/3-\varepsilon}$. In expectation, each vertex has a degree of $n^{1/3-\varepsilon}$ and the number of 6-cycles is $n^{2-6\varepsilon}$. Every vertex in B is connected to u and every vertex in D is connected to u (i.e. $B \cup \{u\}$ and $\{u\} \cup D$ are full bipartite graphs). The example is illustrated in Figure 3.

The number of 3-paths in this graph is at least $n^{2+1/3-\varepsilon}$ while the number of 6-cycles is at most $n^{2-6\varepsilon} + n \cdot n^{4\cdot(\frac{1}{3}-\varepsilon)} = O(n^{2+1/3-4\varepsilon})$. This is because there are $n^{2-6\varepsilon}$ 6-cycles in the bipartite graph $A \cup B$, and every 4-path in $A \cup B$ that starts in B generates a 6-cycle with u and there are at most $n \cdot n^{4\cdot(\frac{1}{3}-\varepsilon)}$ such 4-paths. Notice that there are no other 6-cycles. To conclude, the number of 3-paths is at least $n^{2+1/3-\varepsilon}$ and it is asymptotically bigger than $n^2 + t$ where the number of 6-cycles is $t = O(n^{2+1/3-4\varepsilon})$. Therefore the running time of the algorithm we suggested is lower bounded by $n^{2+1/3-o(1)} + \Omega(t)$.

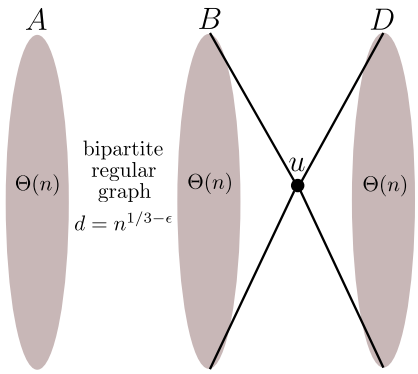


Figure 3 An example of a graph with $O(n^{2+1/3-\varepsilon})$ 3-paths but $O(n^{2+1/3-4\varepsilon})$ 6-cycles. The example shows that 6-cycle listing by listing 3-paths must take $n^{2+1/3-o(1)} + \Omega(t)$ time.

4.2 4-Cycles vs. Triangles: Which is Harder?

As we already mentioned, triangles and 4-cycles are the smallest patterns that are not trivial to find. As such, it is natural to ask which of these patterns are harder to detect. 4-cycle detection can be solved in $O(\min(n^2, m^{\frac{4}{3}}))$ time while triangle detection can be solved in $O(\min(n^\omega, m^{\frac{2\omega}{\omega+1}}))$ time, which is higher as long as $\omega > 2$. This supports the intuition that triangle detection is harder, but this is not formally known. In order to show hardness formally, we have to show a reduction.

▶ **Open Question 18.** *Is there a (fine-grained) reduction from 4-cycle to triangle? Concretely, would an $O(m^{4/3-\varepsilon})$ algorithm for triangle detection (for some $\varepsilon > 0$) imply an $O(m^{4/3-\delta})$ algorithm for 4-cycle detection (for some $\delta = \delta(\varepsilon) > 0$)?*

Even a weak reduction is not known: Suppose that triangle detection is in $O(m)$ time. Does it imply an $o(m^{4/3})$ algorithm for 4-cycle detection?

In an attempt to develop such a reduction we encountered the following barrier: If such a reduction would preserve the number of occurrences (i.e. the number of triangles in the input graph is the same as the number of 4-cycles in the graph generated by the reduction), then this reduction works also for other variations of problems such as *listing* and *counting*.

In this paper, we showed that 4-cycle listing is indeed easier than triangle listing (assuming 3-SUM conjecture). But there is still a gap in the counting problems. The state of the art running time for triangle counting is $O(\min(n^\omega, m^{\frac{2\omega}{\omega+1}}))$ [4] while the best known running time for 4-cycle counting is $O(\min(n^\omega, m^{\frac{4\omega-1}{2\omega+1}}))$ [43, 40]. Under the assumption $\omega = 2$ the running times in the sparse cases are $O(m^{4/3})$ and $O(m^{7/5})$ correspondingly. This means that at least in sparse graphs, 4-cycle counting actually seems harder than triangle counting.

▶ **Open Question 19.** *Is there an algorithm for 4-cycle counting in time $O(m^{\frac{2\omega}{\omega+1}})$? Or alternatively, can we show a hardness result for 4-cycle counting?*

References

1 Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-sum lower bounds for approximate distance oracles via additive combinatorics. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 391–404. ACM, 2023. doi:10.1145/3564246.3585240.

2 Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in p via short cycle removal: cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1487–1500. ACM, 2022. doi:10.1145/3519935.3520066.

3 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. SIAM, 2021.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

5 Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *International Workshop on Computer Science Logic*, pages 208–222. Springer, 2007.

6 R. Bar-Yehuda and S. Even. On approximating a vertex cover for planar graphs. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 303–309, New York, NY, USA, 1982. Association for Computing Machinery. doi:10.1145/800070.802205.

7 Jonathan Berry, Bruce Hendrickson, Randall LaViolette, and Cynthia Phillips. Tolerating the community detection resolution limit with edge weighting. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 83:056119, May 2011. doi:10.1103/PhysRevE.83.056119.

8 Etienne Birmelé, Rui Ferreira, Roberto Grossi, Andrea Marino, Nadia Pisanti, Romeo Rizzi, and Gustavo Sacomoto. Optimal listing of cycles and st-paths in undirected graphs. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1884–1896. SIAM, 2013.

9 Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *International Colloquium on Automata, Languages, and Programming*, pages 223–234. Springer, 2014.

10 Nofar Carmeli and Markus Kröll. On the enumeration complexity of unions of conjunctive queries. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 134–148, 2019.

11 Nofar Carmeli and Markus Kröll. Enumeration complexity of conjunctive queries with functional dependencies. *Theory of Computing Systems*, 64(5):828–860, 2020.

12 Nofar Carmeli and Luc Segoufin. Conjunctive queries with self-joins, towards a fine-grained enumeration complexity analysis. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 277–289, 2023.

13 Keren Censor-Hillel. Distributed subgraph finding: progress and challenges. *arXiv preprint*, 2022. arXiv:2203.06597.

14 Yi-Jun Chang, Seth Pettie, and Hengjie Zhang. Distributed triangle detection via expander decomposition. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 821–840. SIAM, 2019.

15 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985. doi:10.1137/0214017.

16 Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. Finding even cycles faster via capped k-walks. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 112–120, 2017.

17 Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing, 2022. doi:10.48550/arXiv.2210.10173.

18 Arnaud Durand and Etienne Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Transactions on Computational Logic (TOCL)*, 8(4):21–es, 2007.

19 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, 2004.

20 Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. Constant delay algorithms for regular document spanners. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 165–177, 2018.

21 Brooke Foucault Welles, Anne Van Devender, and Noshir Contractor. Is a friend a friend? investigating the structure of friendship networks in virtual worlds. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 4027–4032, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1753846.1754097.

22 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 1–10, 1977.

23 Rajesh Jayaram and John Kallaugher. An optimal algorithm for triangle counting in the stream. *arXiv preprint*, 2021. arXiv:2105.01785.

24 Madhav Jha, C. Seshadhri, and Ali Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 589–597. ACM, 2013. doi:10.1145/2487575.2487678.

25 Tao Jiang and Liana Yepremyan. Supersaturation of even linear cycles in linear hypergraphs. *Combinatorics, Probability and Computing*, 29(5):698–721, 2020. doi:10.1017/S0963548320000206.

26 Ce Jin and Yinzhan Xu. Removing additive structure in 3sum-based reductions. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 405–418. ACM, 2023. doi:10.1145/3564246.3585157.

27 Donald B Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

28 Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3-4):115–121, 2000.

29 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *SODA*, 2016.

30 François Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 216–225. IEEE, 2014.

31 Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007.

32 Dániel Marx and Michał Pilipczuk. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask). *arXiv preprint*, 2013. arXiv:1307.2187.

33 Robert Morris and David Saxton. The number of c2ℓ-free graphs. *Advances in Mathematics*, 298:534–580, 2016. doi:10.1016/j.aim.2016.05.001.

34 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

35 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 603–610, 2010.

36 Dana Richards and Arthur L. Liestman. Finding cycles of a given length. In B.R. Alspach and C.D. Godsil, editors, *Annals of Discrete Mathematics (27): Cycles in Graphs*, volume 115 of *North-Holland Mathematics Studies*, pages 249–255. North-Holland, 1985. doi:10.1016/S0304-0208(08)73019-6.

37 Luc Segoufin. Constant delay enumeration for conjunctive queries. *ACM SIGMOD Record*, 44(1):10–17, 2015.

38 Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977. doi:10.1137/0206036.

39 Virginia Vassilevska Williams, Joshua R Wang, 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*, pages 1671–1680. SIAM, 2014.

40 Virginia Vassilevska Williams, Joshua R. Wang, 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 '15, pages 1671–1680, USA, 2015. Society for Industrial and Applied Mathematics.

41 Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and apsp. *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 786–797, 2020.

42 Raphael Yuster and Uri Zwick. Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222, 1997.

43 Raphael Yuster and Uri Zwick. Detecting short directed cycles using rectangular matrix multiplication and dynamic programming. In *SODA*, volume 4, pages 254–260, 2004.

Monotonicity Characterizations of Regular Languages

Yoav Feinstein
School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

Orna Kupferman
School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

Abstract

Each language $L \subseteq \Sigma^*$ induces an infinite sequence $\{Pr(L,n)\}_{n=1}^{\infty}$, where for all $n \geq 1$, the value $Pr(L,n) \in [0,1]$ is the probability of a word of length n to be in L, assuming a uniform distribution on the letters in Σ. Previous studies of $\{Pr(L,n)\}_{n=1}^{\infty}$ for a regular language L, concerned zero-one laws, density, and accumulation points. We study monotonicity of $\{Pr(L,n)\}_{n=1}^{\infty}$, possibly in the limit. We show that monotonicity may depend on the distribution of letters, study how operations on languages affect monotonicity, and characterize classes of languages for which the sequence is monotonic. We extend the study to languages L of infinite words, where we study the probability of lasso-shaped words to be in L and consider two definitions for $Pr(L,n)$. The first refers to the probability of prefixes of length n to be extended to words in L, and the second to the probability of word w of length n to be such that w^ω is in L. Thus, in the second definition, monotonicity depends not only on the length of w, but also on the words being periodic.

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

Keywords and phrases Regular Languages, Probability, Monotonicity, Automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.26

Funding Supported by the Israel Science Foundation, Grant 2357/19, and by the European Research Council, Advanced Grant ADVANSYNT.

1 Introduction

Consider an alphabet Σ, and assume that letters in Σ are drawn uniformly at random. The probability of a random word of length n to be in a given language $L \subseteq \Sigma^*$ is then $Pr(L,n) = \frac{|\{w \in \Sigma^n | w \in L\}|}{|\Sigma^n|}$. Thus, each language L induces an infinite sequence $\{Pr(L,n)\}_{n=1}^{\infty}$ of values in $[0,1]$. The sequence describes how the length of words influences the probability of their membership in the language.

Several studies in *finite-model theory* refer to the asymptotic behavior of models satisfying a given property. The most known studies in this direction concern *zero-one laws* for different specification formalisms. For example, a zero-one law for first-order sentences states that for every property ψ expressible in first-order logic, the probability of finite structures that are drawn uniformly at random to satisfy ψ tends to 0 or 1 when the size of the structure tends to ∞ [10, 8]. For regular languages, an analogue zero-one law would state that the sequence $Pr(L,n)$ tends to 0 or 1. It is easy to see that regular languages, even unary ones, do not respect a zero-one law. For example, the language $L = (aa)^*$ over $\Sigma = \{a\}$ contains exactly all words of even length, and so $\{Pr(L,n)\}_{n=1}^{\infty}$ alternates between 0 and 1. In [19], Sin'ya characterized regular languages whose asymptotic probability converges to 0 or 1, and described a linear-time algorithm for deciding whether the language of a given deterministic automaton has a zero-one behavior.

Another study of the asymptotic behavior of $\{Pr(L,n)\}_{n=1}^{\infty}$ concerns *accumulation points* of the sequence, namely points to which a subsequence converges to. As shown in [2, 18], when L is regular, there are only finitely many such points, and they are all rational. The

© Yoav Feinstein and Orna Kupferman;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 26; pp. 26:1–26:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

above works also study the *density* of L, which is the limit of $\{Pr(L,n)\}_{n=1}^{\infty}$. A Markov-chain based approach to reasoning about the density of a regular language is presented in [4], which describes a cubic algorithm for calculating the density (or determine that one does not exist) of a language given by a deterministic automaton. Finally, the limit of the sequence $\frac{1}{n}\sum_{i\in\{1,\ldots,n\}} Pr(L,i)$, is studied in [3], and its convergence is related to that of $\{Pr(L,n)\}_{n=1}^{\infty}$.

In this paper we study the *monotonicity* of $\{Pr(L,n)\}_{n=1}^{\infty}$. We say that L is *eventually monotonic* if there is $m \geq 1$ such that for all $n \geq m$, we have that $Pr(L,n+1) \geq Pr(L,n)$ (monotonically non-decreasing) or for all $n \geq m$, we have that $Pr(L,n+1) \leq Pr(L,n)$ (monotonically non-increasing). When $m = 1$, the sequence is *monotonic*, and when the probability is strictly increased or decreased, we say that the sequence is monotonically increasing or monotonically decreasing. Let us consider again the language $L = (aa)^*$, yet assume it is defined over the alphabet $\Sigma = \{a,b\}$. Now, $\{Pr(L,n)\}_{n=1}^{\infty}$ does tend to 0. Indeed, $Pr(L,n) = \frac{1}{2^n}$ for even n's, and is 0 for odd n's. On the other hand, $\{Pr(L,n)\}_{n=1}^{\infty}$ is *never monotonic*, as for every $n \geq 1$, we have that $Pr(L,2n) > Pr(L,2n+1)$ yet $Pr(L,2n+1) < Pr(L,2n+2)$. Thus, a language L may have a zero-one behavior and not be eventually monotonic. Implication in the other direction does not hold either. For example, the language $L = b\cdot(a+b)^* + a^*$ has $Pr(L,n) = \frac{1}{2} + \frac{1}{2^n}$. Thus, L is monotonically decreasing, yet it tends to $\frac{1}{2}$, and does not have a zero-one behavior.

We start with some theoretical properties of monotonicity of regular languages. Recall that we define $Pr(L,n)$ with respect to a uniform distribution on the alphabet. We show that the probability according to which letters are drawn may actually change the monotonicity characteristic of languages, even in the limit. This is in contrast with zero-one laws for regular languages, which are independent of the distribution (as long as all letters have a positive probability). We study the sensitivity of monotonicity to Boolean operations. It is easy to come up with languages with dual monotonicity whose union and intersections are not monotonic. We show that the union and intersection of languages that are both increasing or both decreasing need not be monotonic, even eventually. We then consider the case of unary languages, thus when $\Sigma = \{a\}$. There, $Pr(L,n) \in \{0,1\}$, and it is easy to characterize monotonic languages. We continue and point to positive cases, namely classes of monotonic languages. For example, we show that $\{Pr(L,n)\}_{n=1}^{\infty}$ is eventually monotonic when L is recognizable by a *deterministic weak automaton* of *depth* or *width* 1. The characterization is tight, in the sense that removing one of these limitations, one may end up in a language that is never monotonic. A different tight characterization we give is of 2-state *counter-free* deterministic automata. Our analysis is based on results from linear algebra regarding the *stochastic matrix* induced by the automata, and it is valid also with respect to non-uniform distributions of the alphabet.

Moving to languages of infinite words, we consider lasso-shaped words, and study three sequences. The first two, denoted $\{Pr^{\exists}(L,n)\}_{n=1}^{\infty}$ and $\{Pr^{\forall}(L,n)\}_{n=1}^{\infty}$, refer to the probability of prefixes of length n to be extendable, by some or all suffixes, respectively, to words in L. We show that $Pr^{\exists}(L,n)$ and $Pr^{\forall}(L,n)$ are related to the probability of words of length n to be good prefixes for L and bad prefixes for the complement of L, respectively, implying the monotonicity of the sequences. The third sequence is $\{Pr^{\omega}(L,n)\}_{n=1}^{\infty}$, where $Pr^{\omega}(L,n)$ is the probability of a word w of length n to be such that w^{ω} is in L. Thus, here, monotonicity depends also on the words being periodic. For example, while it is easy to see that the language of finite prefixes of $(aab)^*$ is monotonically decreasing, with $Pr(L,n) = \frac{1}{2^n}$, we have that $Pr^{\omega}((aab)^{\omega},n)$ is never monotonic, as it is 0 for n's that are not multiplications of 3.

We describe a construction that transforms a deterministic parity automaton $\mathcal{A}$ to a deterministic automaton for all finite words u such that $u^\omega \in L(\mathcal{A})$. The construction is exponential, and we prove a matching lower bound. Using the construction, we are able to lift some of the positive results about languages of finite words to the setting of infinite words. We also discuss the characterizations when the languages are given by formulas in LTL (or LTL_f, for the case of finite words) [15, 9].

Beyond the theoretical interest, properties of $\{Pr(L,n)\}_{n=1}^{\infty}$ are useful when reasoning about L. We give here some examples. In the context of decision problems about regular languages, researchers suggested approximated algorithms that refer to asymptotic behavior. For example, [14] studies *almost equivalence* of regular languages, where two languages $L_1, L_2 \subseteq \Sigma^*$ are almost equivalent if $Pr(L_1 \triangle L_2, n)$ tends to 0, where $L_1 \triangle L_2$ denotes the symmetric difference between L_1 and L_2. General monotonicity properties of $L_1 \triangle L_2$ indicate how L_1 and L_2 differ from each other in the limit. Similar reasoning can be made about intersection of languages, their union, and more.

Our study of monotonicity for languages of infinite words is motivated by the need to sort results of vacuity checks in model checking. *Vacuity* detection is a method for finding errors in the model-checking process when the specification is found to hold in the model. Most vacuity algorithms are based on checking the effect of applying mutations on the specification [1]. It has been recognized that while in many cases vacuity results are valued as highly informative, there are also cases in which the results are viewed as meaningless by users [17, 5]. In [7], the authors suggested to rank vacuity results based on the probability of the mutated specification to hold in a random computation. For example, two natural mutations of the specification $G(req \rightarrow Fready)$ are $G(\neg req)$ and $GFready$. It is agreed that satisfaction of the first mutation is more alarming than satisfaction of the second. The methodology explains this by the probability of $G(\neg req)$ to hold in a random computation being 0, whereas the probability of $GFready$ being 1. The above definition assumes an infinite computation. As discussed in [7], in the context of model checking, it is more relevant to refer to the probability of the mutation to hold in computations in finite-state systems. Specifically, it is suggested in [7] to study $Pr(L,k,l)$, which is the probability that a lasso-shaped word with a prefix of length k and a loop of length l, belongs to L. Our study of lasso-shaped words does this for the special cases $l = \infty$ or $k = 0$. In particular, our study of $Pr^\omega(L,l)$ addresses challenges that concern the periodic nature of lasso-shaped words and shows that the periodicity of lasso-shaped words affects their probability to be in L in ways that are orthogonal and not not less significant than their length.

2 Preliminaries

2.1 Automata

An *alphabet* Σ is a finite set of letters. A *word* over Σ is a finite or infinite sequence $w = \sigma_1, \sigma_2, \sigma_3, \cdots$ of letters from Σ. We use $|w|$ to denote the length of w, with $|w| = \infty$ for an infinite word w. We use Σ^* and Σ^ω to denote the set of all finite and infinite words over Σ, respectively. A *language* is a set of words. For a language $L \subseteq \Sigma^*$, we use $comp(L)$ to denote the language complementing L, thus $comp(L) = \Sigma^* \backslash L$.

A *nondeterministic automaton* is $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$, where Σ is a finite input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function, $Q_0 \subseteq Q$ is a set of initial states, and $\alpha \subseteq Q$ is a set of accepting states. We extend δ to finite words in the expected way. Thus, $\delta^*(q,u)$ is the state $\mathcal{A}$ reaches when it reads the word $u \in \Sigma^*$ from some state $q \in Q$. Formally, $\delta^* : Q \times \Sigma^* \rightarrow Q$ is such that for every $q \in Q$, we have that $\delta^*(q,\epsilon) = q$ and for a finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $\delta^*(q, u \cdot \sigma) = \delta(\delta^*(q,u), \sigma)$.

A *run* of $\mathcal{A}$ on a word w is the function $r : \{0 \leq i \leq |w|\} \to Q$, such that $r(0) \in Q_0$, and for all $i \geq 0$, we have that $r(i+1) \in \delta(r(i), \sigma_{i+1})$. If $|Q_0| = 1$, and for all $q \in Q$ and $\sigma \in \Sigma$, it holds that $|\delta(q, \sigma)| = 1$, then $\mathcal{A}$ has a single run on w, and we say that $\mathcal{A}$ is *deterministic*. We sometimes refer to a run also as a sequence of states; that is, $r = r(0), r(1), \ldots \in Q^{|w|+1}$.

When $\mathcal{A}$ runs on finite words, the run r is finite, and it is accepting iff it ends in an accepting state, thus $r(|w|) \in \alpha$. When $\mathcal{A}$ runs on infinite words, acceptance depends on the set $\inf(r)$, of the states that r visits infinitely often. Formally $\inf(r) = \{q \in Q :$ for infinitely many $i \in \mathbb{N}$, we have that $r(i) = q\}$. As Q is finite, the set $\inf(r)$ is guaranteed not to be empty. In *Büchi* automata, the run r is accepting iff r visits the set of accepting states infinitely often, thus $\inf(r) \cap \alpha \neq \emptyset$. Otherwise, r is rejecting. In *parity* automata, the acceptance condition is $\alpha : Q \to \{1, \ldots, k\}$, and a run r is accepting iff the minimal *color* in $\{1, \ldots, k\}$ that r visits infinitely often is even. The automaton $\mathcal{A}$ accepts a word w if there exists an accepting run r of $\mathcal{A}$ on w. The *language* of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We also say that $\mathcal{A}$ recognizes $L(\mathcal{A})$. We define the size of $\mathcal{A}$, denoted $|\mathcal{A}|$, as the number of states that $\mathcal{A}$ has. A deterministic automaton $\mathcal{A}$ on finite words is *minimal* if there is no equivalent automaton, namely one that accepts the same language, of a smaller size.

We use NFW and DFW to denote nondeterministic and deterministic automata on finite words, respectively, and similarly for NBW, DBW, NPW, and DPW, denoting Büchi and parity automata.

An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ is *weak* if there exists a partition of Q into sets $Q_1, Q_2, \ldots, Q_k$ such that for all $1 \leq i \leq k$, either $Q_i \subseteq \alpha$, in which case we say that Q_i is accepting, or $Q_i \cap \alpha = \emptyset$, in which case we say that Q_i is rejecting. In addition, there is a partial order $\leq$ on the sets such that transitions in δ lead to states of the same or of lower sets. Formally, for all states $q, q' \in Q$, if $q' \in \delta(q, \sigma)$, for some letter $\sigma \in \Sigma$, then the sets Q_i and Q_j for which $q \in Q_i$ and $q' \in Q_j$ satisfy $Q_j \leq Q_i$. Equivalently, the partition into strongly connected components of the graph induced by $\mathcal{A}$ is such that each component is either contained in α or disjoint from α. For $j \in \mathbb{N}$, we say that $\mathcal{A}$ is *weak*$[j]$ if $\mathcal{A}$ is weak and the largest set Q_i in the partition of Q is of size j. We refer to j as the *width* of $\mathcal{A}$. Describing classes of weak$[j]$ automata, we add $[j]$ to the acronym. For example an NFW[1] is a weak[1] NFW, namely an NFW in which all cycles are self loops. We sometimes refer also to the *depth* of $\mathcal{A}$, which is the maximal number of alternations between accepting and rejecting sets that a run may have.

A regular language L is *counter-free* (*CF*, for short) if there is $n \geq 1$ such that for every $m \geq n$ and $v, w, x \in \Sigma^*$, we have that $vw^n x \in L$ iff $vw^m x \in L$. For example, $L_1 = (ab)^*$ is CF, while $L_2 = (aa)^*$ is not CF. An NFW $\mathcal{A}$ is CF if $L(\mathcal{A})$ is CF. A DFW $\mathcal{A}$ is *permutation-free* (*PF*, for short) if there does not exist a non-trivial permutation between its states. That is, there does not exist a word $w \in \Sigma^*$ and a set $\{q_1, q_2, \ldots, q_l\}$, with $l \geq 2$ of different states in $\mathcal{A}$ such that $\delta^*(q_i, w) = q_{(i \mod l)+1}$ for all $1 \leq i \leq l$. A regular language is PF if its minimal automaton is PF. By [13] (Theorem 5.1), a regular language is CF iff its minimal DFW is PF.

A language $L \subseteq \Sigma^*$ is a *safety* language if every word not in L has a bad prefix. Formally, if $w \notin L$, then w has a prefix $x \in \Sigma^*$ such that for every $y \in \Sigma^*$, we have that $x \cdot y \notin L$. For example, if $\Sigma = \{a, b\}$, then the languages $L_1 = a^*$ is safety. Indeed, if $w \notin L_1$, then w has a prefix $x \in a^* \cdot b$, and $x \cdot y \notin L_1$ for all $y \in \Sigma^*$. On the other hand, the language $L_2 = (ab)^*$ is not safety, as, for example, the word aba is not in L_3, yet every prefix of it can be extended to a word in L_3. A language $L \subseteq \Sigma^*$ is a *co-safety* language if *comp*(L) is safety. Equivalently, every word $w \in L$ has a good prefix, namely a prefix x such that $x \cdot y \in L$ for

all $y \in \Sigma^*$. Safety and co-safety languages can be recognized by weak automata of depth 1 [20, 12]. Indeed, in automata for safety languages, all runs start in accepting sets and move to a rejecting sink once a bad prefix is read. Dually, for co-safety languages, run start in rejecting sets and may move to an accepting sink.

The definitions of safety and co-safety languages apply also for languages $L \subseteq \Sigma^\omega$. Here, a bad prefix is $x \in \Sigma^*$ such that for every $y \in \Sigma^\omega$, we have that $x \cdot y \notin L$. Note that while the language $L_2 = (ab)^*$ of finite words is not safety, the language $L_2' = (ab)^\omega$ of infinite words is safety. Indeed, $w \notin L_2'$ iff w has a prefix ending with aa or bb, which is a bad prefix.

2.2 Monotonicity Characterizations of Regular Languages

Let $\{a_n\}_{n=1}^\infty = a_1, a_2, a_3, \ldots$ be some sequence of real numbers in $[0, 1]$. For convenience, we sometimes write $\{a_n\} = \{a_n\}_{n=1}^\infty$ for short. We say that the sequence $\{a_n\}_{n=1}^\infty$ is:

- *monotonically non-decreasing* (MND) if for all $n \geq 0$, we have that $a_{n+1} \geq a_n$. If for all $n \geq 0$, we have that $a_{n+1} > a_n$, then the sequence is *monotonically increasing* (MI).
- *monotonically non-increasing* (MNI) if for all $n \geq 0$, we have that $a_{n+1} \leq a_n$. If for all $n \geq 0$, we have that $a_{n+1} < a_n$, then the sequence is *monotonically decreasing* (MD).

We say that the sequence $\{a_n\}$ is *monotonic* (M) if $\{a_n\}$ is MNI or MND. Since we care about limit behavior, we have particular interest in sequences that are not immediately monotonic but rather monotonic from a certain index. For $\gamma \in \{\text{MND}, \text{MI}, \text{MNI}, \text{MD}, \text{M}\}$, a sequence $\{a_n\}$ is *eventually* γ, if there exists $k \geq 0$, such that $\{a_n\}_{n=k}^\infty$ is γ. If $\{a_n\}$ is not EM, we say that it is *never-monotonic* (NM). If the sequence $\{a_n\}$ is not EM, we say that it is *never-monotonic* (NM). We refer to $\gamma \in \{$MND, MI ,MNI, MD, M, EMND, EMI, EMNI, EMD, EM, NM$\}$ as the *monotonicity characterization* of languages. We use $\tilde{\gamma}$ is the monotonicity characterization dual to γ. Thus, $\{a_n\}_{n=1}^\infty$ is γ iff $\{1 - a_n\}_{n=1}^\infty$ is $\tilde{\gamma}$. For example, $\widetilde{\text{MI}} = \text{MD}$.

Consider an alphabet Σ. We assume that letters in Σ are drawn uniformly at random (see Section 3.1 for an extension to arbitrary distributions). Accordingly, the probability of each letter to be drawn is $\frac{1}{|\Sigma|}$ and for a given length $n \geq 1$, the probability of a word of length n to be drawn is $\frac{1}{|\Sigma^n|}$. Consider a language $L \subseteq \Sigma^*$. The probability of a random word of length n to be in L is then $Pr(L, n) = \frac{|\{w \in \Sigma^n | w \in L\}|}{|\Sigma^n|}$. We characterize regular languages by the way the length of words influences membership in the language. For a language $L \subseteq \Sigma^*$ and a monotonicity characterization γ, we say that L is γ iff the sequence $\{Pr(L, n)\}_{n=1}^\infty$ is γ. Likewise, an automaton $\mathcal{A}$ is γ iff $L(\mathcal{A})$ is γ. Note that we start with $n = 1$ and ignore the membership of ϵ in L.

▶ **Example 1.** Consider the following languages over $\Sigma = \{a, b\}$.

- The language $L_1 = a^*$ induces the sequence $\{Pr(L_1, n) = \frac{1}{2^n}\}$, and is therefore MD.
- Its complement language $L_2 = \Sigma^* \cdot b \cdot \Sigma^*$ induces the sequence $\{Pr(L_2, n) = 1 - \frac{1}{2^n}\}$, and is therefore MI.
- Let $L_3 = (\epsilon + (\Sigma^* \cdot a)) \cdot (bb)^*$. Thus, L_3 contains exactly all words in which the number of b's after the last occurrence of a is even. The first elements in $\{Pr(L_3, n)\}$ are described in the table below

n	1	2	3	4
$Pr(L_3, n)$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{5}{8}$	$\frac{11}{16}$

For example, when $n = 3$, then out of 8 words of length 3, only the 5 words aaa, baa, bba, aba, and abb are in L_3. Although we can already determine that $\{Pr(L_3, n)\}$ is not M, it might be EM. ⌟

▶ **Example 2.** Let $S_1, S_2 \subset \Sigma$ be a partition of an alphabet Σ into two nonempty sets. Consider the language $L_{once} = S_1^* \cdot S_2 \cdot S_1^*$. Thus, a word w is in L iff it contains exactly one occurrence of a letter in S_2. On the one hand, longer words are more likely to include a letter in S_2. On the other hand, longer words are more likely to include more than one such letter. Formally, if $p = \frac{|S_2|}{|\Sigma|}$ is the probability of a random letter to be in S_2, then it is not hard to show that $Pr(L_{once}, n) = n \cdot p \cdot (1-p)^{n-1}$. Thus, L_{once} is EM, yet monotonicity only starts when $n \geq \frac{1}{-\ln(1-p)}$. ⌟

3 Theoretical Properties

In this section we study some theoretical properties of monotonicity for regular languages. Recall that we define $Pr(L, n)$ as the probability of a word of length n to be in L, assuming that letters are drawn uniformly at random. In Section 3.1, we show that the probability according to which letters are drawn may actually change the monotonicity characteristic of languages, even in the limit. In Section 3.2, we study the sensitivity of the monotonicity to operations like complementation, union, and intersection. Finally, in Section 3.3, we consider the case of unary languages, thus when $\Sigma = \{a\}$ and $Pr(L, n) \in \{0, 1\}$ for all $n \geq 1$.

3.1 On the choice of a uniform distribution

Some properties, like a zero-one behavior of a language, are independent of the distribution of the letters in the alphabet [19]. In Theorem 3, we show that for monotonicity, the distribution may actually affect the characterization. Moreover, the distribution may not only turn monotonic languages into be eventually monotonic ones or turn strict monotonicity (that is, MI or MD) into non-strict one (that is, MND or MNI), but may turn a strictly monotonic language into one that is never monotonic:

▶ **Theorem 3.** *Let $\Sigma = \{a, b\}$. There is a language L such that L is MD when the letters in Σ are uniformly distributed and is NM in all distributions $f : \Sigma \to [0, 1]$ with $f^2(a) > f(b)$.*

Proof. Consider the language $L = aa(aa)^* + b(aa)^*$. Note that all the words in L are of the form $\sigma \cdot a^m$, with σ inducing the required parity of m: if $\sigma = a$, then m has to be odd, and if $\sigma = b$, then m has to even. Thus, for every length $n \geq 1$, there is exactly one word of length n in L. If n is even, then it is the word a^n, and if n is odd, then it is the word $b \cdot a^{n-1}$.

Accordingly, it is not hard to see that under a uniform distribution, we have that $Pr(L, n) = \frac{1}{2^n}$, and so L is MD. On the other hand, consider a distribution $f : \Sigma \to [0, 1]$ with $f(a) = p_a$ and $f(b) = p_b$. Then, $Pr_f(L, n) = p_a^n$ when n is even, and $Pr_f(L, n) = p_b \cdot p_a^{n-1}$ when n is odd. Note that when n is odd, then $Pr_f(L, n+1) = p_b \cdot Pr_f(L, n)$, thus $Pr_f(L, n+1) < Pr_f(L, n)$. In addition, $Pr_f(L, n+2) = p_a^2 \cdot Pr_f(L, n)$. Thus, if $p_a^2 > p_b$, then $Pr_f(L, n+2) > Pr_f(L, n+1)$. Thus, when $p_a^2 > p_b$, the sequence $\{Pr_f(L, n)\}_{n=1}^{\infty}$ is NM. For example, when $p_a = \frac{3}{4}$ and $f(b) = \frac{1}{4}$, we get that $Pr_f(L, n) = \frac{3^n(2+(-1)^n)}{4^n \cdot 3}$. ◀

Theorem 3 may question the robustness of our results. As we argue in Remark 20, however, all the results in the paper apply to arbitrary distributions.

3.2 Monotonicity characterization and Boolean operations

In this section we study whether the monotonicity characterization of languages is preserved under complementation, union, and intersection.

We start with complementation. Since for every language L, and for every $n \geq 1$, we have that $Pr(comp(L), n) = 1 - Pr(L, n)$, dualization follows immediately from the definitions:

▶ **Theorem 4.** *For every language $L \subseteq \Sigma^*$ and monotonicity characterization γ, we have that L is γ iff $comp(L)$ is $\tilde{\gamma}$.*

We continue to union and intersection. It is easy to come up with languages with dual monotonicity characterization whose union and intersections are not monotonic. We show that, surprisingly, also the union and intersection of languages that are both monotonically increasing or both monotonically decreasing, need not be monotonic, even in the limit.

▶ **Theorem 5.** *The intersection and union of MD (or MI) languages may be NM.*

Proof. We prove the result for MD languages. By considering the complement languages, one can get proofs for MI languages.

We start with intersection. Consider the languages $L_1 = a^*$ and $L_2 = (aa)^* + b(bb)^*$. It is not hard to show that for every $n \geq 0$, we have that $Pr(L_1, n) = Pr(L_2, n) = \frac{1}{2^n}$. Thus, both languages are MD. Let $L_\cap = L_1 \cap L_2$. Note that $L_\cap = (aa)^*$. Accordingly, $Pr(L_\cap, n)$ is 0 for odd n's and is $\frac{1}{2^n}$ for even n's. Hence, $L_\cap$ is NM.

We continue with union. Consider the languages $L_3 = a\Sigma^* + b^*$ and $L_4 = a\Sigma(\Sigma\Sigma)^* + (bb)^* + b(\Sigma\Sigma)^* + a(aa)^*$. It is not hard to see that for every $n \geq 1$, we have that, $Pr(L_3, n) = Pr(L_4, n) = \frac{1}{2} + \frac{1}{2^n}$. Thus, both languages are MD. Let $L_\cup = L_3 \cup L_4$. Note that $L_\cup = a\Sigma^* + (bb)^* + \Sigma(\Sigma\Sigma)^*$. Thus, $Pr(L_\cup, n)$ is 1 for odd n's and is $\frac{1}{2} + \frac{1}{2^n}$ for even n's. Hence, $L_\cup$ is NM. ◀

3.3 The case of unary languages

In this section we examine languages over a unary alphabet, thus when $\Sigma = \{a\}$. Note that then, each language $L \subseteq a^*$ corresponds to a set of integers, namely all $n \geq 0$ such that $a^n \in L$. Accordingly, $Pr(L, n) = 1$ if $a^n \in L$, and $Pr(L, n) = 0$ if $a^n \notin L$. It follows that a unary language cannot be MI or MD, and that MND and MNI unary languages are trivial. Indeed, only $L = a^*$ and $L = \emptyset$ have $Pr(L, n) = 1$ and $Pr(L, n) = 0$ for all $n \geq 0$, respectively. Thus, a unary language $L \subseteq a^*$ can have one of the following two monotonicity characteristics.

- EM, which holds if L is *almost trivial*, thus L or $comp(L)$ are finite. Formally, there is $m \geq 0$ such that either $a^k \in L$ for all $k \geq m$, or $a^k \notin L$ for all $k \geq m$.
- NM if for all $k \geq 0$, we have $m, l \geq k$ with $a^m \in L$ and $a^l \notin L$.

Note that the definitions coincide with these presented in Section 2.2, simplified to the case $Pr(L, n)$ is in $\{0, 1\}$ for all $n \geq 1$.

▶ **Theorem 6.** *Unary NFW[1]s are EM.*

Proof. Consider an NFW[1] $\mathcal{A} = \langle \{a\}, Q, \delta, Q_0, \alpha \rangle$ with m states. We distinguish between two cases. First, if $\mathcal{A}$ has a state $q \in \alpha$ that is reachable from Q_0 by a path that includes a state with a self loop (possibly q itself has a self loop), then $\{a^k : k \geq m\} \subseteq L(\mathcal{A})$, and so $\{Pr(L(\mathcal{A}), n)\}_{n=1}^{\infty}$ is eventually always 1. Otherwise, all the states in α are reachable only by paths that do not include states with self loops. Thus, $|L(\mathcal{A})| \leq |\alpha|$, and so $\{Pr(L(\mathcal{A}), n)\}_{n=1}^{\infty}$ is eventually always 0. ◀

Note that the graph induced by a unary DFW $\mathcal{A}$ that accepts an infinite language must be *lasso-shaped*. Also, $\mathcal{A}$ is EM iff all the states in the cycle of the lasso are accepting or all are not accepting. Accordingly, arbitrary unary DFWs may be NM, yet all weak unary DFWs are EM. In the nondeterministic setting, the characterization in Theorem 6 is tight, and even unary NFW[2]s may be NM:

▶ **Theorem 7.** *Unary NFW[2]s may be NM.*

Proof. Consider the NFW[2] appearing in Figure 1. It is not hard to see that its language is $(aa)^+$, which is NM. ◀

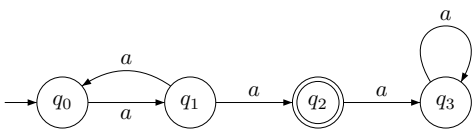


Figure 1 A unary NFW[2] whose language is $(aa)^+$.

4 An Algebraic Approach

In this section we relate monotonicity characterization with properties of the stochastic matrix that describes the behavior of a given DFW. We first need some definitions.

A *Markov chain* is a tuple $M_{\mathcal{A}} = \langle Q, \tau \rangle$, where Q is a set of states and $\tau : Q \times Q \to [0, 1]$ is a probabilistic transition function: for $q, q' \in Q$, the value $\tau(q, q')$ is the probability of moving from q to q'. Since τ describes a distribution, then for every $q \in Q$, we have that $\sum_{s \in Q} \tau(q, s) = 1$.

Each DFW $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ induces a Markov chain $M_{\mathcal{A}} = \langle Q, \tau \rangle$, where for $q, q' \in Q$, we have that $\tau(q, q')$ is the probability of a run that visits state q to move to state q' when it reads the next letter. Thus, assuming a uniform distribution on Σ, we have that $\tau(q, q') = \frac{|\{\sigma : \delta(q,\sigma) = q'\}|}{|\Sigma|}$.

▶ **Example 8.** Consider the DFW $\mathcal{A}$ appearing in Figure 2. Its induced Markov chain $M_{\mathcal{A}}$ appears to its right.

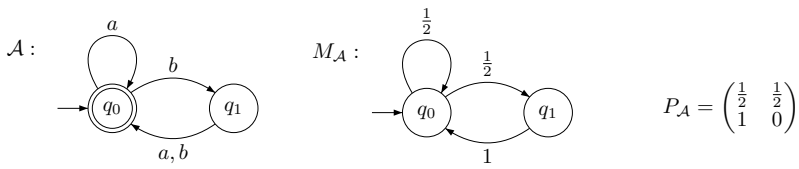


Figure 2 A DFW $\mathcal{A}$, its induced Markov chain $M_{\mathcal{A}}$, and its stochastic matrix $P_{\mathcal{A}}$.

Note that $\mathcal{A}$ recognizes the language L_3 discussed in Example 1, namely the language of all words that have an even number of b's after the last occurrence of a. In Example 1, we described the first elements in $\{Pr(L_3, n)\}$. In particular, we calculated $Pr(L_3, 3)$ by counting the number of words of length 3 that are in L_3. We now show how to calculate $Pr(L_3, 3)$ by examining the Markov chain $M_{\mathcal{A}}$. A word of length 3 is accepted by $\mathcal{A}$ only if its run on $\mathcal{A}$ is $q_0q_0q_0q_0$, which happens with probability $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2}$, or $q_0q_1q_0q_0$, which happens with probability $\frac{1}{2} \cdot 1 \cdot \frac{1}{2}$, or $q_0q_0q_1q_0$, which happens with probability $\frac{1}{2} \cdot \frac{1}{2} \cdot 1$. Overall, we have that $Pr(L(\mathcal{A}), 3) = \frac{5}{8}$. ⌟

The *stochastic matrix* of a DFW $\mathcal{A}$, denoted $P_{\mathcal{A}}$, describes the transition function τ of its Markov chain $M_{\mathcal{A}}$. Formally, we assume some order on the states in Q, thus $Q = \{q_1, q_2, \ldots, q_n\}$, and $P_{\mathcal{A}}$ is an $n \times n$ matrix with elements in $[0, 1]$, where for $1 \leq i, j \leq m$ we have that $P_{\mathcal{A}_{i,j}} = \tau(q_i, q_j)$. Given a DFW $\mathcal{A}$ and states $q_i, q_j \in Q$, the probability of a run on a word of length n that starts in q_i to end in the state q_j is $(P^n_{\mathcal{A}})_{i,j}$. Let x^n_i denote the probability of a word of length n to end in state q_i. If the initial state of $\mathcal{A}$ is q_1, then $x^n_i = (P^m_{\mathcal{A}})_{1,i}$.

▶ **Example 9.** Back to the DFW $\mathcal{A}$ from Figure 2. The matrix $P_{\mathcal{A}}$ appears on the right of the figure. Recall that x_0^n and x_0^n denote the probability of a word to end on the states q_0 and q_1, respectively. By diagonalizing $P_{\mathcal{A}}$ (see details in Appendix A), we can calculate:

$$(x_0^n, x_1^n) = (1,0) \cdot P_{\mathcal{A}}^n = \left(\frac{2 + (\frac{-1}{2})^n}{3}, \frac{1 - (\frac{-1}{2})^n}{3} \right).$$

Since $\alpha = \{q_0\}$, we have that $Pr(L(\mathcal{A}), n) = x_0^n = \frac{2+(\frac{-1}{2})^n}{3}$. Note that $Pr(L(\mathcal{A}), n) > Pr(L(\mathcal{A}), n-1)$, when n is even, and $Pr(L(\mathcal{A}), n) < Pr(L(\mathcal{A}), n-1)$, when n is odd. Hence, $L(\mathcal{A})$ is NM. ⌟

An *eigenvalue* of matrix A is a scalar, denoted λ_i, such that there exists a non-zero vector v for which $Av = \lambda_i v$. In Appendix A, we describe how to reason about $(P_{\mathcal{A}}^m)_{1,i}$ using known results from linear algebra. Specifically, we prove the following lemma.

▶ **Lemma 10.** *Consider a stochastic matrix $P \in \mathbb{R}^{d \times d}$ and a set $S = \{(i_1, j_1), \cdots, (i_t, j_t)\}$ of t pairs of indices in P. If all the eigenvalues of P are real and non-negative, then the sequence $\{\sum_{(i',j') \in S} P_{i',j'}^n\}_{n=1}^{\infty}$ is EM.*

Lemma 10 is useful for reasoning about the monotonicity of DFWs.

▶ **Theorem 11.** *Let $\mathcal{A}$ be some DFW. If $P_{\mathcal{A}}$ has only real and non-negative eigenvalues, then $L(\mathcal{A})$ is EM.*

Proof. Let $q_{l_1}, q_{l_2}, \ldots, q_{l_t}$ be the accepting states of $\mathcal{A}$, and let q_s be its initial state. Let $S = \{(s, l_1), (s, l_2), \ldots, (s, l_t)\}$. As $\mathcal{A}$ is deterministic, then $Pr(L(\mathcal{A}), n) = \sum_{(i',j') \in S} P_{i',j'}^n$. By Lemma 10, we thus have that $\{Pr(L, n)\}$ is EM. ◀

5 Classes of Monotonic Languages

In this section we point to three classes of monotonic languages. The first two refer to weak automata and the third to permutation-free automata. The classes are tight, in the sense that if we remove some limitations in their definition, we end up in classes that are never monotonic.

5.1 Weak automata – width

We start with the width of weak automata and show that if the sets in the partition that witnesses the weakness of the automaton are singletons, then its language is EM, and that this sufficient condition is tight.

▶ **Theorem 12.** *DFW[1]s are EM.*

Proof. Let $\mathcal{A}$ be some DFW[1] and let $\{q_0\} \leq \{q_1\} \leq \ldots \leq \{q_{m-1}\}$ be the partition of the states of $\mathcal{A}$ into sets that witness its weakness. Then, for every $i < j$, we have that $\tau(q_i, q_j) = 0$, implying that $P_{\mathcal{A}}$ is an upper triangular matrix, and so its eigenvalues are on its diagonal. Hence, as the elements of $P_{\mathcal{A}}$ are real and non-negative, so are it eigenvalues. Thus, by Theorem 11, we have that $\mathcal{A}$ is EM. ◀

▶ **Example 13.** Recall the language L_{once} from Example 2. Figure 3 describes a DFW[1] for L_{once}, which immediately implies it is EM. ⌟

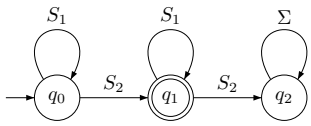


Figure 3 A DFW[1] for L_{once}.

We now show that Theorem 12 is tight, in the sense we cannot remove any of the limitations imposed by the DFW[1] restrictions. Thus, DFW[2]s and NFW[1]s need not be monotonic. Note that this is in contrast with the unary case, where NFW[1]s are EM (Theorem 6).

▶ **Theorem 14.** *DFW[2]s may be NM.*

Proof. Consider the DFW[2] $\mathcal{D}$ that appears in Figure 4 (left). It is not hard to prove that $L(\mathcal{D}) = a(ba)^*a$. Thus, $\mathcal{D}$ only accept words of even length. Moreover, while the probability of words of odd length to be accepted is 0, the probability is positive for words of even length. Specifically, $Pr(L(\mathcal{D}), n) = \frac{(-1)^n+1}{2^{n+1}}$, which is NM. ◀

▶ **Theorem 15.** *NFW[1]s may be NM.*

Proof. Let $\Sigma = \{a, b\}$ and consider the NFW $\mathcal{A}$ appearing in Figure 4 (right). It is not hard to see that $L(\mathcal{A}) = comp((ab)^*)$. Indeed, $comp((ab)^*) = \Sigma^* \cdot a + b \cdot \Sigma^* + \Sigma^* \cdot (aa + bb) \cdot \Sigma^*$. Since $(ab)^*$ is infinite and contains only words of even length, it is NM. Hence, by Theorem 4, so is $L(\mathcal{A})$. ◀

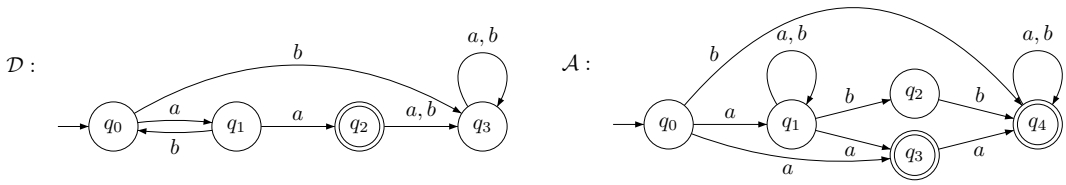


Figure 4 A DFW[2] recognizing $a(ba)^*a$ and an NFW[1] recognizing $comp((ab)^*)$.

5.2 Weak automata – depth

We continue to the depth of weak automata, namely the bound on the number of alternations between accepting and rejecting sets in the partition that witnesses the weakness. Weak automata of depth 1 recognize safety and co-safety languages [20, 12]. Intuitively, the longer a word is, the more likely it is for a bad thing to do happen. Formally, we have the following.

▶ **Theorem 16.** *All safety (co-safety) languages of finite words are MNI (MND, respectively).*

Proof. We prove that all co-safety languages are MND. The result for safety follows from Theorem 4. For each $n \geq 0$, we calculate the number $\#(L, n)$ of words in $L \cap \Sigma^n$. First $\#(L, 0) \in \{0, 1\}$, according to the membership of ϵ in L. Then, counting words in $L \cap \Sigma^{n+1}$, observe that since L is co-safety, then each word in $L \cap \Sigma^n$ contributes to $L \cap \Sigma^{n+1}$ all its $|\Sigma|$ extensions by one letter. Thus, $\#(L, n+1) \geq \#(L, n) \cdot |\Sigma|$. Since $\#(L, n) = Pr(L, n) \cdot |\Sigma|^n$, we get that

$$Pr(L, n+1) \geq \frac{Pr(L, n) \cdot |\Sigma|^n \cdot |\Sigma|}{|\Sigma|^{n+1}} = Pr(L, n).$$ ◀

Recall the DFW[2] $\mathcal{D}$ and NFW[1] $\mathcal{A}$ that are described in Figure 4. In Theorem 14, we proved that their languages are NM. As $\mathcal{D}$ has depth 2 and $\mathcal{A}$ has depth 1, they also serve for proving the tightness of Theorem 16. Thus, we have the following.

▶ **Theorem 17.** *All weak DFWs of depth* 1 *are M. On the other hand, there is a weak DFW of depth* 2 *and a weak NFW of depth* 1 *whose languages are NM.*

5.3 Permutation-free automata

We conclude with automata that recognize CF languages, and give a tight sufficient condition also for them.

▶ **Theorem 18.** *2-state CF DFWs are M.*

Proof. In Figure 5, we describe a general 2-state DFW $\mathcal{A} = \langle \Sigma, \{q_0, q_1\}, \delta, q_0, \alpha \rangle$, with its stochastic matrix $P_{\mathcal{A}}$. If both q_0, q_1 are accepting or both are rejecting, then $Pr(L(\mathcal{A}), n) = 1$ or 0 respectively. We assume that exactly one of the states of $\mathcal{A}$ is accepting. Let $\tau(q_0, q_0) = p_0$ and $\tau(q_1, q_1) = p_1$, where τ is the transition function of the Markov chain $M_{\mathcal{A}}$. Consider the stochastic matrix $P_{\mathcal{A}}$ of $\mathcal{A}$, shown in Figure 5. The eigenvalues of $P_{\mathcal{A}}$ can be calculated and are $\lambda_1 = 1$ and $\lambda_2 = p_0 + p_1 - 1$ (note that λ_2 might be equal to 1 as well). Note that $Pr(L(\mathcal{A}), n) = c_1 + c_2(p_0 + p_1 - 1)^n$, for some $c_1, c_2 \in \mathbb{R}$. We show that $p_0 + p_1 - 1 \geq 0$, implying that $\mathcal{A}$ is M.

Assume by way of contradiction that $p_0 + p_1 - 1 < 0$. Then $(1 - p_0) + (1 - p_1) > 1$. Therefore, there exists some $\sigma \in \Sigma$, such that $\delta(q_0, \sigma) = q_1$ and $\delta(q_1, \sigma) = q_0$, and so there is a non-trivial permutation in $\mathcal{A}$. Since only one of the states of $\mathcal{A}$ is accepting, it is minimal. Thus, $\mathcal{A}$ is PF, and we have reached a contradiction. ◀

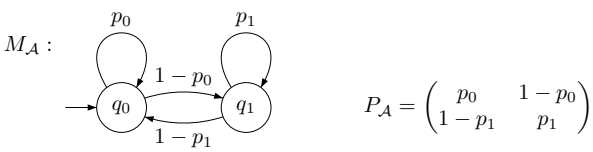


Figure 5 A CF DFW $\mathcal{A}$ with two states.

▶ **Theorem 19.** *2-state DFWs, 3-state CF DFWs, and 2-state CF NFWs may be NM.*

Proof. First, in Example 9, we saw that the 2-state DFW from Example 8 is NM. For a 3-state CF DFW, consider the DFW $\mathcal{D}$ appearing in Figure 6. It is not hard to prove that $\mathcal{D}$ is PF and that $L(\mathcal{D}) = comp((ab)^*)$, which is NM. Now, for a 2-state CF NFW, consider automaton $\mathcal{A}$ appearing to the right of $\mathcal{D}$. Note that $L(\mathcal{A}) = L(\mathcal{D})$, Indeed, $\mathcal{D}$ can be obtained by applying the subset construction on $\mathcal{A}$. Thus, $\mathcal{A}$ is a 2-state CF NFW whose language is NM. ◀

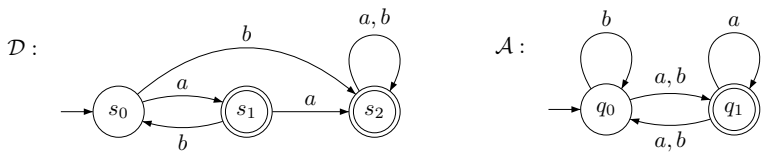


Figure 6 A 3-state DFW and a 2-state NFW for the PF and NM language $comp((ab)^*)$.

▶ Remark 20. It is not hard to check that all the positive results in this section can be extended to nonuniform distribution of the letters in the alphabet. Indeed, the proofs of Theorems 12 and 18 are independent of the uniform distributions, and the proof of Theorem 16 can be easily adjusted to the general case. ⌟

▶ Remark 21. The logic LTL_f is a linear temporal logic that specifies languages of finite words. By [6], PF DFWs are as expressive as LTL_f. In particular, the LTL_f formula $p \wedge ((p \wedge X\neg p) \vee (\neg p \wedge Xp))U(p \wedge X(p \wedge last))$ is satisfied only by computations of even length, and is similar to the NM DFW[2] from Figure 4. ⌟

6 Infinite Words

For languages of infinite words, we consider lasso-shaped words, thus words of the form vu^ω, for some $v, u \in \Sigma^*$. We study three sequences. The first two refer to the prefix of the lasso, namely to the probability of prefixes $v \in \Sigma^n$ to be extendable, by some or all suffixes, to words in L. The third refers to the loop of the lasso, namely to the probability of a loop $u \in \Sigma^n$ to be such that $u^\omega \in L$. We relate the three sequences to sequences induced by languages of finite words. While this is straightforward for the two sequences that refer to the prefix, it involves new ideas and constructions for the third sequence, where reasoning depends not only on the length of the lasso, but also on the words being periodic.

6.1 Monotonicity in the length of the prefix

For a language $L \subseteq \Sigma^\omega$, let $good.pref^\exists(L) = \{v \in \Sigma^* \mid \text{there is } u \in \Sigma^\omega \text{ such that } v \cdot u \in L\}$ and $good.pref^\forall(L) = \{v \in \Sigma^* \mid \text{ for all } u \in \Sigma^\omega \text{ we have that } v \cdot u \in L\}$. We define $Pr^\exists(L,n) = \frac{|good.pref^\exists(L) \cap \Sigma^n|}{|\Sigma^n|}$ and $Pr^\forall(L,n) = \frac{|good.pref^\forall(L) \cap \Sigma^n|}{|\Sigma^n|}$. Thus, $\{Pr^\exists(L,n)\}_{n=1}^\infty$ and $\{Pr^\forall(L,n)\}_{n=1}^\infty$ describe how the length of words influences their probability to be extendable to words in L by some or by all suffixes, respectively. Intuitively, the longer a prefix is, the smaller is the probability to extend it to a word in L by some suffix, yet the higher is the probability that all its extensions result in a word in L. Formally, we have the following.

▶ **Theorem 22.** *For every language $L \subseteq \Sigma^\omega$, we have that $\{Pr^\exists(L,n)\}_{n=1}^\infty$ is MNI and $\{Pr^\forall(L,n)\}_{n=1}^\infty$ is MND.*

Proof. It is easy to see that $good.pref^\exists(L)$ is a safety language. Indeed, if $v \in \Sigma^*$ cannot be extended to a word in L, then so do all its extensions. Also, as $good.pref^\forall(L) = comp(good.pref^\exists(comp(L)))$, we have that $good.pref^\forall(L)$ is a co-safety language.

By definition, $Pr^\exists(L,n) = Pr(good.pref^\exists(L), n)$ and $Pr^\exists(L,n) = Pr(good.pref^\forall(L), n)$. Thus, the claim follows from Theorem 16. ◀

6.2 Monotonicity in the length of the loop

We continue and study the sequence$\{Pr^\omega(L,n)\}_{n=1}^\infty$. Recall that $Pr^\omega(L,n)$ is the probability of a word $u \in \Sigma^n$ to be such that $u^\omega \in L$. Formally, $Pr^\omega(L,n) = \frac{|\{u \in \Sigma^n | u^\omega \in L\}|}{|\Sigma^n|}$. Thus, here, in addition to the dependency of membership in L in the length of u, monotonicity may depend on the periodic nature of words of the form u^ω.

In order to illustrate this dependency, we start with a simple observation.

▶ **Theorem 23.** *Safety and co-safety languages of infinite words may be NM.*

Proof. Consider the safety language $L = (ab)^\omega$. Note that for all $n \geq 0$, we have that $Pr^\omega(L, 2n) > 0$, whereas $Pr^\omega(L, 2n+1) = 0$. Indeed, for $u = (ab)^n$, we have that $u^\omega \in L$, whereas for u of an odd length, either u contains aa or bb, or the first and last letters of u are identical. In both cases, $u^\omega \notin L$. Dually, $comp(L)$ is a co-safety language, and for all $n \geq 0$ we have that $Pr^\omega(L, 2n+1) = 1$, whereas $Pr^\omega(L, 2n) < 1$. ◀

On the other hand, it is not hard to extend the results in Section 3.2 about complementation, union, and intersection, also to languages on infinite words.

▶ **Theorem 24.** *For every language $L \subseteq \Sigma^\omega$ and monotonicity characterization γ, we have that L is γ iff $comp(L)$ is $\tilde{\gamma}$. On the other hand, the intersection and union of MD (or MI) languages of infinite words may be NM.*

Proof. The proof for complementation is identical to the one in the case of finite words. Indeed, also in the case of infinite words, we have that $Pr^\omega(comp(L), n) = \frac{|\{u \in \Sigma^n | u^\omega \notin L\}|}{|\Sigma^n|} = 1 - \frac{|\{u \in \Sigma^n | u^\omega \in L\}|}{|\Sigma^n|} = 1 - Pr^\omega(L, n)$.

For union and intersection, we prove the result for MD languages over the alphabet $\Sigma = \{a, b, c, d\}$. We start with intersection. Consider the languages $L_1 = a^\omega + (ab)^\omega$ and $L_2 = b^\omega + (ab)^\omega$. Observe that a finite word $w \in \Sigma^*$ is such that $w^\omega \in L_1$ iff $w = a^*$ or $w = (ab)^*$. Therefore, $Pr(L_1, n)$ is $\frac{1}{4^n}$ for odd n's, and is $\frac{2}{4^n}$ for even n's. Since for every $n \geq 1$, we have that $\frac{2}{4^n} > \frac{1}{4^{n+1}}$ and $\frac{1}{4^n} > \frac{2}{4^{n+1}}$, it follows that L_1 is MD. A similar argument implies that L_2 is MD. Let $L_\cap = L_1 \cap L_2$. Note that $L_\cap = (ab)^\omega$. Thus, $w^\omega \in L_\cap$ iff $w = (ab)^*$, and so $L_\cap$ is NM.

For union, consider the MD languages $L_3 = a^\omega + (ab)^\omega + (ac)^\omega$ and $L_4 = a^\omega + (ba)^\omega + (bc)^\omega$. Let $L_\cup = L_3 \cup L_4$. Note that $L_\cup = a^\omega + (ab)^\omega + (ac)^\omega + (ba)^\omega + (bc)^\omega$. It is not hard to show that $Pr(L_\cup, n)$ is $\frac{1}{4^n}$ for odd n's, and is $\frac{5}{4^n}$ for even n's. Since $Pr(L(\mathcal{A}), n) > Pr(L(\mathcal{A}), n+1)$ for odd n's, and $Pr(L(\mathcal{A}), n) < Pr(L(\mathcal{A}), n+1)$ for even n's, we have that $L_\cup$ is NM. ◀

We continue and examine CF languages of infinite words. Recall that for finite words, we showed that every 2-state CF DFW is EM, and that the characterization is tight. By going over all 2-state NBWs, one can show that they are all M. In fact, the latter holds regardless of the distribution on the letters. On the other hand, the 3-state DBW for $comp((ab)^\omega)$ is CF and NM. Hence, we have the following.

▶ **Theorem 25.** *2-states NBWs are M, yet 3-state CF NBWs may be NM.*

Proof. Consider the 2-state NBW $\mathcal{A} = \langle \Sigma, \{q_0, q_1\}, q_0, \delta, \alpha \rangle$. If both states are accepting or rejecting, then $Pr(L(\mathcal{A}), n) = 1$ or 0 respectively. Let $\Sigma_{i \to j} = \{\sigma \in \Sigma \mid \delta(q_i, \sigma) = q_j\}$. If $\alpha = \{q_0\}$, then a finite word $w \in \Sigma^*$ is such that $w^\omega \notin L(\mathcal{A})$ if every run of w reaches the state q_1 and there does not exist a letter σ in w such that $\sigma \in \Sigma_{1 \to 0}$. Let f be some distribution on Σ. The probability that all the runs on w^ω, for a random word w of length n to be rejecting, under a distribution f, is exactly $(\sum_{\sigma \in \Sigma \setminus \Sigma_{1 \to 0}} f(\sigma))^{n-1} \cdot (\sum_{\sigma \in \Sigma_{0 \to 1} \cap \Sigma \setminus (\Sigma_{1 \to 0} \cup \Sigma_{0 \to 0})} f(\sigma))$. Since $\sum_{\sigma \in \Sigma \setminus \Sigma_{1 \to 0}} f(\sigma) \geq 0$, we have that $\mathcal{A}$ is M.

If $\alpha = \{q_1\}$, then a finite word $w \in \Sigma^*$ is such that $w^\omega \notin L(\mathcal{A})$ if every run of w never visits q_1. The probability that random word w of length n to be such that $w^\omega \notin L(\mathcal{A})$, under a distribution f, is then $(\sum_{\sigma \in \Sigma \setminus \Sigma_{0 \to 1}} f(\sigma))^n$. Hence, $\mathcal{A}$ is M. ◀

Note that all LTL formulas induce CF languages [21]. For example, the infinite counterpart of the language from Example 2 is $L^\omega_{once} = S_1^* \cdot S_2 \cdot S_1^\omega$, which corresponds to the formula $S_1 U (S_2 \wedge XG(S_1))$. Note that for all $n \geq 1$, we have that $Pr^\omega(L^\omega_{once}, n) = 0$, as a periodic word includes either none or infinitely many occurrences of S_2. On the other hand, for the

language L induced by the formula $GF(S_2)$, we have $Pr^\omega(L,n) = 1 - \frac{1}{2^n}$, which is strictly monotonic. Indeed, L can be recognized by a 2-state NBW. A more surprising example is $L' = GF(S_2 \wedge X(S_2))$, which requires infinitely many successive occurrences of S_2. Note that the length of the required sequence of S_2's is even, and so the periodic nature of words may hint that L' is not monotonic. However, as L' can be recognized by a 2-state NBW, Theorem 25 implies that it is actually monotonic.

We continue to our most technically elaborated result. For $L \subseteq \Sigma^\omega$, let $L^{\frac{1}{\omega}} = \{u \in \Sigma^* : u^\omega \in L\}$. We describe a construction that enables us to reduce reasoning about $Pr^\omega(L,n)$ to reasoning about $Pr(L^{\frac{1}{\omega}}, n)$. In Theorem 26 we describe the construction for L given by a DPW, which involves an exponential blow-up. Then, in Theorem 27, we prove an exponential lower bound for the construction, in fact for a family of languages definable by DBWs of depth 1.

▶ **Theorem 26.** *For every DPW $\mathcal{A}$, there exists a DFW $\mathcal{A}^{\frac{1}{\omega}}$ such that $L(\mathcal{A}^{\frac{1}{\omega}}) = L(\mathcal{A})^{\frac{1}{\omega}}$. The construction of $\mathcal{A}^{\frac{1}{\omega}}$ is effective and its size is exponential in the size of $\mathcal{A}$.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, \alpha \rangle$, with $\alpha : Q \to \{1, \ldots, k\}$. For two functions $s : Q \to Q$ and $c : Q \to \{1, \ldots, k\}$, and a word $u \in \Sigma^*$, we say that the pair $\langle s, c \rangle$ *captures* u if for all states $q \in Q$, we have that $s(q) = \delta^*(q, u)$, and $c(q) = \min\{\alpha(\delta^*(q, v)) : v \text{ is a prefix of } u\}$. Thus, s describes how each of the states in Q proceeds when it reads u, and c describes the minimum color that is visited along the way. Note that since $\mathcal{A}$ is deterministic, then every word u has a single pair of functions that captures it. We combine s and c to a single function $f : Q \to Q \times [k]$, and define successive applications of f by $f^0(q) = \langle q, \alpha(q) \rangle$ and $f^{i+1}(q) = \langle \delta^*(s^i(q), u), \min\{c^i(q), c(s^i(q)\} \rangle$. Thus, $f^i(q) = \langle q', j \rangle$, for $q' = \delta^*(q, u^i)$ and j being the minimum color that is visited when u^i is read from q.

Let $f : Q \to Q \times [k]$ be a function that captures a word $u \in \Sigma^*$, and let $f = \langle s, c \rangle$. By the pigeonhole principle, there must be $0 \le i_1 < i_2 \le |Q|$ for which $s^{i_1}(q_0) = s^{i_2}(q_0)$. Let $i_1 < i_2$ be such indices that minimize i_2, and let q be the state such that $s^{i_1}(q_0) = s^{i_2}(q_0) = q$. By the definition of acceptance in parity automata, we have that u^ω is accepted from q_0 iff $c^{i_2 - i_1}(q)$ is even. Indeed, the run from q_0 on u^ω starts by reaching q after reading u^{i_1}, and then repeatedly traverses a loop in q while reading $u^{i_2 - i_1}$, with $c^{i_2 - i_1}(q)$ being the minimal color that is visited along the loop, making it the minimal color visited infinitely often while reading u^ω. Accordingly, we say that a function $f : Q \to Q \times [k]$ is *good* if the minimal indices $0 \le i_1 < i_2$ for which $s^{i_1}(q_0) = s^{i_2}(q_0)$ are such that $c^{i_2 - i_1}(s^{i_1}(q_0))$ is even. Note that the definition of f is independent of a word. Yet, a word $u \in \Sigma^\omega$ is accepted by $\mathcal{A}$ iff the function f that captures it is good. Also note that deciding whether a given function f is good can be done in polynomial time.

We define the DFA $\mathcal{A}^{\frac{1}{\omega}} = \langle \Sigma, \mathcal{F}, \rho, f_0, \alpha' \rangle$, where $\mathcal{F}$ is the set of functions $f : Q \to Q \times \{1, \ldots, k\}$, and f_0 and ρ are defined so that after reading a word $u \in \Sigma^*$, the DFA reaches a state that captures u. Formally, the initial state f_0 is the function that captures ϵ, thus $s_0(q) = q$ and $c_0(q) = \alpha(q)$, for all $q \in Q$, and the transitions function ρ is such that for every state $\langle s, c \rangle \in \mathcal{F}$ and letter $\sigma \in \Sigma$, we have that $\rho(\langle s, c \rangle, \sigma) = \langle s', c' \rangle$, where for every $q \in Q$, we have that $s'(q) = \delta(s(q), \sigma)$ and $c'(q) = \min\{c(q), \alpha(s'(q))\}$. Finally, α' is the set of all good functions. Since a word $u \in \Sigma^\omega$ is accepted by $\mathcal{A}$ iff the function f that captures it is good, we have that $L(\mathcal{A}^{\frac{1}{\omega}}) = L(\mathcal{A})^{\frac{1}{\omega}}$, and we are done. ◀

▶ **Theorem 27.** *There is a family $L_1, L_2, L_3, \ldots$ of languages of infinite words such that for every $n \ge 1$, there is a DBW of depth 1 with $O(n)$ states that recognizes L_n, yet every DFW that recognizes $L_n^{\frac{1}{\omega}}$ needs at least 2^n states.*

Proof. For $n \geq 1$, let $\Sigma_n = \{1, 2, \ldots, n\}$. We define

$$L_n = \{\Sigma_n^* \cdot \# \cdot \sigma_1 \cdot \sigma_2 \cdots \sigma_k \cdot \# \cdot (\Sigma_n + \#)^\omega : \sigma_1 \in \{\sigma_2, \ldots, \sigma_k\}\}.$$

Note that $L_n \subseteq (\Sigma_n \cup \{\#\})^\omega$. First, a DBW for L_n waits for the first #, records in its state space the letter σ after it, then waits for σ to appear before a second #, after which the DBW moves to an accepting sink. If $\sigma = \#$ or if the second # appears before σ reappears, the DBW goes to a rejecting sink. Clearly, the DBW needs only $O(n)$ states. Also, each word in L_n has a good prefix, and so L_n is a co-safety language, and the DBW is of depth 1.

Assume by way of contradiction that there is a DFW $\mathcal{A}_n = \langle \Sigma_n, Q, q_0, \delta, \alpha \rangle$ that recognizes $L_n^{\frac{1}{\omega}}$ and has less than 2^n states. For a set $S \subseteq \Sigma_n$, let $w_S \in \Sigma_n^*$ be a word that contains exactly all the letters in S. Since $\mathcal{A}_n$ has less than 2^n states, there are two sets $S_1, S_2 \subseteq \Sigma_n^*$ such that the words w_{S_1} and w_{S_2} lead to the same state in $\mathcal{A}_n$. Formally, $\delta^*(q_0, w_{S_1}) = \delta^*(q_0, w_{S_2}) = q$, for some state q. Let $\sigma \in \Sigma_n^*$ be such that, without loss of generality, $\sigma \in S_1 \setminus S_2$. Consider the state $q' = \delta^*(q, \sigma \cdot \#)$. On the one hand, as $(w_{S_2} \cdot \sigma \cdot \#)^\omega \notin L_n$, we have that $q' \notin \alpha$. On the other hand, as $(w_{S_1} \cdot \sigma \cdot \#)^\omega \in L_n$, and $\mathcal{A}_n$ is deterministic, we have that $q' \in \alpha$, and we have reached a contradiction. ◀

In the context of monotonicity, the construction of $\mathcal{A}^{\frac{1}{\omega}}$ enables us to lift the positive result about DFW[1]s to DBW[1]s. For this, we prove that the construction in Theorem 26 preserves weak[1]ness:

▶ **Theorem 28.** *DBW[1]s are EM.*

Proof. Consider a DBW[1] $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. In order to use the notations in Theorem 26, we view α as a parity condition $\alpha : Q \to \{1, 2, 3\}$. Let $\mathcal{A}^{\frac{1}{\omega}} = \langle \Sigma, \mathcal{F}, \rho, f_0, \alpha' \rangle$ be the DFW obtained by applying the construction in Theorem 26 on $\mathcal{A}$. We prove that $\mathcal{A}^{\frac{1}{\omega}}$ is a DFW[1]. Then, EMness of $L(\mathcal{A})$ follows from the fact that $Pr^\omega(L(\mathcal{A}), n) = Pr(L(\mathcal{A}^{\frac{1}{\omega}}, n)$ and Theorem 12.

Consider a reachable state $f \in \mathcal{F}$. Let $u \in \Sigma^*$ be such that $\rho^*(f_0, u) = f$. Thus, f captures u. Consider a letter $\sigma \in \Sigma$, and let $f' = \rho(f, \sigma)$. Thus, f' captures $u \cdot \sigma$. We prove that if $f \neq f'$, then $f \neq \rho^*(f, \sigma \cdot w)$, for all $w \in \Sigma^*$. Thus, the only cycles through the state f are self loops.

Let $f = \langle s, c \rangle$, $f' = \langle s', c' \rangle$, and assume that $f' \neq f$. First, if for all states $q \in Q$, we have that $s(q) = s'(q)$, then, as $c'(q) = \min\{c(q), \alpha(s'(q))\}$, it must be that for all states $q \in Q$, we also have that $c(q) = c'(q)$. Thus, $f' \neq f$ implies that there is a state $q \in Q$ such that $s(q) \neq s'(q)$. Since f captures u and f' captures $u \cdot \sigma$, the latter implies that $\delta^*(q_0, u) \neq \delta^*(q_0, u \cdot \sigma)$. Hence, as $\mathcal{A}$ is a DBW[1], we have that $\delta^*(q_0, u) \neq \delta^*(q_0, u \cdot \sigma \cdot w)$ for all $w \in \Sigma^*$. Thus, $f \neq \rho(f, \sigma \cdot w)$, for all $w \in \Sigma^*$, and we are done. ◀

We now show that Theorem 28 is tight.

▶ **Theorem 29.** *DBW[2]s and NBW[1]s may be NM.*

Proof. Consider the DBW[2] obtained by viewing the DFW from Figure 6 (left) as a Büchi automaton with $\alpha = \{s_2\}$. Consider also the NBW[1] obtained by viewing the NFW[1] appearing in Figure 4 (right) as a Büchi automaton. It is easy to see that both Büchi automata recognize the language $comp((ab)^\omega)$. As argued in the proof of Theorem 23, the language is NM. ◀

7 Discussion

We studied how the length of words influences their probability to belong to a regular language. We characterized formalisms that induce monotonic languages. The characterization is tight, in the sense that all the restrictions composing it are essential. Nevertheless, the characterization is not a necessary condition, in the sense that there are languages that do not satisfy it and are monotonic. The general problem of deciding the monotonicity characteristics of the language of a given automaton is the subject of future research. As discussed in Section 3.3, the question is very easy for unary automata. Moreover, our construction of $\mathcal{A}^{\frac{1}{\omega}}$ solves the challenges that have to do with periodicity and extends a solution for the setting of finite words to a solution for the setting of infinite words. We showed that the answer to the question depends also in the distribution of the letters in the alphabet. Since the problem of finding the spectrum of a stochastic matrix is difficult, our problem is expected to be difficult too. A related problem is to find the length in which eventually monotonic languages start to be monotonic. As we have seen in Example 2, this distance may be exponential in the size of the automaton also for deterministic weak automata of width 1.

References

1. I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh. Efficient detection of vacuity in ACTL formulas. *Formal Methods in System Design*, 18(2):141–162, 2001.
2. J. Berstel. Sur la densité asymptotique de langages formels. In *Proc. 1st Int. Colloq. on Automata, Languages, and Programming*, pages 345–358, 1972.
3. J. Berstel, D. Perrin, and C. Reutenauer. *Codes and Automata*, volume 129 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2010.
4. M. Bodirsky, T. Gärtner, T. von Oertzen, and J. Schwinghammer. Efficiently computing the density of regular languages. In *Proc. 6th Latin American Symposium on Theoretical Informatics*, volume 2976 of *Lecture Notes in Computer Science*, pages 262–270. Springer, 2004.
5. M. Chechik, M. Gheorghiu, and A. Gurfinkel. Finding state solutions to temporal queries. In *Proc. Integrated Formal Methods*, 2007. To appear.
6. J. Cohen, D. Perrin, and J-Eric Pin. On the expressive power of temporal logic. *Journal of Computer and System Sciences*, 46(3):271–294, 1993.
7. S. Ben David and O. Kupferman. A framework for ranking vacuity results. In *11th Int. Symp. on Automated Technology for Verification and Analysis*, volume 8172 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2013.
8. R. Fagin. Probabilities in finite models. *Journal of Symb. Logic*, 41(1):50–5, 1976.
9. G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 854–860, 2013.
10. Y.V. Glebskii, D.I. Kogan, M.I. Liogonkii, and V.A. Talanov. Range and degree of realizability of formulas in the restricted predicate calculus. *Kibernetika*, 2:17–28, 1969.
11. Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.
12. O. Kupferman and M.Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.
13. R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Pres, 1971.
14. Y. Nakamura. The almost equivalence by asymptotic probabilities for regular languages and its computational complexities. In *Proc. 7th International Symposium on Games, Automata, Logics and Formal Verification*, volume 226 of *EPTCS*, pages 272–286, 2016.
15. A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

16 Jeffrey S. Rosenthal. Convergence rates for markov chains. *SIAM Review*, 37(3):387–405, 1995.

17 Ben-David S, D. Fisman, and S. Ruah. Temporal antecedent failure: Refining vacuity. In *Proc. 18th Int. Conf. on Concurrency Theory*, volume 4703 of *Lecture Notes in Computer Science*, pages 492–506. Springer, 2007.

18 A. Salomaa and M. Soittola. *Automata Theoretic Aspects of Formal Power Series*. Springer-Verlag, 1978.

19 R. Sin'ya. An automata theoretic approach to the zero-one law for regular languages: Algorithmic and logical aspects. In *Proc. 6th International Symposium on Games, Automata, Logics and Formal Verification*, volume 193 of *EPTCS*, pages 172–185, 2015.

20 A.P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6:495–511, 1994.

21 P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

A Useful Results from Linear Algebra

In this section we elaborate on the relevant results from linear algebra that are used in the proof of Theorem 11. The considerations are similar to these developed in [16]. For simplicity, we assume that all matrices we consider are squared (unless stated otherwise).

Recall that an *eigenvalue* of a matrix A is a scalar (denoted as λ_i) such that there exists a non-zero vector v for which $Av = \lambda_i v$.

The *characteristic polynomial* of A is the equation $det(A - \lambda I)$ where I is the identity matrix, *det* represent the determinant operation and λ is a scalar.

It is known that the eigenvalues of A are the roots of the characteristic polynomial (see [11], Chapter 1).

The *algebraic multiplicity* of an eigenvalue λ_i is its multiplicity as a root of the characteristic polynomial.

For two matrices A and B, we say that A is *similar* to matrix B if there exists an invertible matrix U such that $A = UBU^{-1}$. It can be shown that similar matrices have the same characteristic polynomials, as such they have the same eigenvalues.

A matrix is *diagonal* if all its non-zero elements are on the diagonal, and is *diagonalizable* if it is similar to some diagonal matrix.

A *Jordan block* is a matrix A with some $\lambda \in \mathbb{C}$ on the diagonal and 1's on the superdiagonal, thus A is of the following form

$$\begin{pmatrix} \lambda & 1 & 0 & 0 \\ 0 & \lambda & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & \lambda \end{pmatrix}.$$

The 1×1 matrix (λ) is also a Jordan block. A Jordan Block with $\lambda \in \mathbb{C}$ on its diagonal is sometimes called a *Jordan block of* λ.

A *Jordan matrix* is a matrix with Jordan blocks on its diagonal. In other words, let $\{B_1, \ldots, B_n\}$ be some Jordan blocks. Then, a Jordan matrix is of the following form

$$\begin{pmatrix} B_1 & 0 & 0 & 0 \\ 0 & B_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & B_n \end{pmatrix}.$$

Let A be some matrix. It is known that although not all matrices are diagonalizable, there always exists a Jordan matrix J that is similar to A (see [11], Chapter 3). Furthermore, the Jordan blocks of J are made of the eigenvalues of A and each block is of size that is not bigger than the algebraic multiplicity of its eigenvalue. Matrix J is called the *Jordan normal form* of A. Note that J is only unique up to the order of its Jordan blocks.

Jordan blocks give us a way to calculate the n'th power of a matrix directly. Let B be a Jordan block of size k. Then, for $n > k$, we have that

$$B^n = \begin{pmatrix} \lambda^n & \binom{n}{1}\lambda^{n-1} & \binom{n}{2}\lambda^{n-2} & \cdots & \cdots & \binom{n}{k-1}\lambda^{n-k+1} \\ 0 & \lambda^n & \binom{n}{1}\lambda^{n-1} & \cdots & \cdots & \binom{n}{k-2}\lambda^{n-k+2} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \lambda^n & \binom{n}{1}\lambda^{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & \lambda^n \end{pmatrix}. \tag{1}$$

▶ **Example 30.** Recall that in Example 9 we considered the DFW $\mathcal{A}$ from Example 8 and calculated the probabilities x_0^n and x_1^n of a run of length n to end on the states q_0 and q_1, respectively. We can now elaborate on the calculation, which is based on diagonalizing $P_{\mathcal{A}}$:

$$P_{\mathcal{A}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{-1}{2} & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{-1}{2} & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{-2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix}$$

Accordingly,

$$\begin{aligned} \begin{pmatrix} x_0^n & x_1^n \end{pmatrix} &= \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot P_{\mathcal{A}}^n \\ &= \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \frac{-1}{2} & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{-1}{2} & 0 \\ 0 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} \frac{-2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix} \\ &= \begin{pmatrix} \frac{2+\left(\frac{-1}{2}\right)^n}{3} & \frac{1-\left(\frac{-1}{2}\right)^n}{3} \end{pmatrix}. \end{aligned} \tag{2}$$

⌟

Let $M \in \mathbb{R}^{d \times d}$ be a square matrix. Let $\lambda_1, \lambda_2, \ldots, \lambda_l$ be the distinct eigenvalues of M in descending order by their absolute value. That is, $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_l|$. If M is diagonalizable, then for every index (i, j) in M, we have that

$$(M^n)_{ij} = c_1(\lambda_1)^n + c_2(\lambda_2)^n + \cdots + c_l(\lambda_l)^n, \tag{3}$$

where $c_1, c_2, \ldots, c_l \in \mathbb{C}$ are complex numbers that depend on (i, j). If M is not diagonalizable, then we instead rely on the Jordan normal form of M, which always exists. Using Property (1), where $n > d$, we replace each expression $c_i(\lambda_i)^n$ in (3) by

$$(c_i^1(n)^{k_i-1} + c_i^2(n)^{k_i-2} + \cdots + c_i^{k_i})(\lambda_i)^n,$$

where k_i is the size of the largest Jordan block of λ_i, and $c_i^1, c_i^2, \ldots, c_i^{k_i} \in \mathbb{C}$.

By [11] (Theorem 3.1.11), if all the eigenvalues of M are real, then $c_1^1, \ldots, c_1^{k_1}, \ldots, c_l^1, \ldots, c_l^{k_l}$ are real too.

Let $P \in \mathbb{R}^{d \times d}$ be some stochastic matrix. It is shown in [16] that for all $1 \leq i \leq l$, we have that $|\lambda_i| \leq 1$, and that $\lambda_1 = 1$ is an eigenvalue of P. Furthermore, the largest Jordan block of $\lambda_1 = 1$ is of size 1.

We can now prove Lemma 10, which is key to the proof of Theorem 11.

▶ **Lemma 10.** *Consider the stochastic matrix $P \in \mathbb{R}^{d\times d}$ and the set $S = \{(i_1, j_1), \cdots, (i_t, j_t)\}$ of t indices in P. If all the eigenvalues of P are real and non-negative, then the sequence $\{\sum_{(i',j')\in S} P^n_{i',j'}\}_{n=1}^{\infty}$ is EM.*

Proof. Let $\lambda_1, \lambda_2, \ldots, \lambda_l$ be the distinct eigenvalues of P in descending order by their absolute value. Let $\{a_n\} = \{\sum_{(i',j')\in S} P^t_{i',j'}\}_{t=1}^{\infty}$. Since P is a stochastic matrix, then for every $1 \le i \le l$, we have that $|\lambda_i| \le 1$. By our assumption, all the eigenvalues are real and non-negative, and so, for every $1 \le i \le l$, we have that $0 \le \lambda_i \le 1$. Since the largest Jordan block of $\lambda_1 = 1$ is of size 1, then for every $n > d$, we get that

$$a_n = (1)^n c^1_1 + (\lambda_2)^n((n)^{k_2-1}c^1_2 + (n)^{k_2-2}c^2_2 + \cdots + c^{k_2}_2) + \cdots + (\lambda_l)^n((n)^{k_l-1}c^1_l + \cdots + c^{k_l}_l),$$

for some $c^1_1, c^1_2, \ldots, c^{k_2}_2, c^1_3, \ldots, c^1_l, \ldots, c^{k_l}_l \in \mathbb{R}$.

Now, observe that when $n > d$, the difference between two successive iterations of the sequence is

$$a_n - a_{n+1} = (\lambda_2)^n((n)^{k_2-1}(c^1_2 - c^1_2\lambda_2) + \cdots + c^{k_2}_2 - c^{k_2}_2\lambda_2) + (\lambda_3)^n(\cdots) + \cdots + (\lambda_l)^n(\cdots) \quad (4)$$

If Equation (4) is equal to zero, we have that $a_n = a_{n+1}$ and so $\{a_n\}$ is EM. Else, it is not hard to prove that since $1 > \lambda_2 > \lambda_3 > \cdots > \lambda_l \ge 0$, then there exists some non-zero element $(\lambda_i)^n(n)^{k_i-j}(c^j_i - c^j_i\lambda_i)$ that is dominant in Equation (4). That is, there exist $m_0 \ge d$ such that for all $m \ge m_0$, we have that

$$|(\lambda_i)^n(n)^{k_i-j}(c^j_i - c^j_i\lambda_i)| \ge |a_n - a_{n+1} - (\lambda_i)^n(n)^{k_i-j}(c^j_i - c^j_i\lambda_i)|. \quad (5)$$

Note that since $\lambda_i > 0$, we have that $(\lambda_i)^n(n)^{k_i-j} > 0$. Since c^j_i can be any real number, we need to distinguish between cases. If $c^j_i - c^j_i\lambda_i > 0$, then $a_n - a_{n+1} > 0$, implying that $\{a_n\}_{m_0}^{\infty}$ is MD. If $c^j_i - c^j_i\lambda_i < 0$, we have that $a_n < a_{n+1}$, in which case $\{a_n\}_{m_0}^{\infty}$ is MI. In both cases, $\{a_n\}$ is EM. ◀

Note that Example 8 does not contradict Theorem 11, as $\lambda = -\frac{1}{2}$ is an eigenvalue of P.

Decision Tree Complexity Versus Block Sensitivity and Degree

Rahul Chugh
Citadel Securities, London, UK

Supartha Podder
Stony Brook University, NY, USA

Swagato Sanyal
IIT Kharagpur, India

Abstract

Relations between the decision tree complexity and various other complexity measures of Boolean functions is a thriving topic of research in computational complexity. While decision tree complexity is long known to be polynomially related with many other measures, the optimal exponents of many of these relations are not known. It is known that decision tree complexity is bounded above by the cube of block sensitivity, and the cube of polynomial degree. However, the widest separation between decision tree complexity and each of block sensitivity and degree that is witnessed by known Boolean functions is quadratic.

Proving quadratic relations between these measures would resolve several open questions in decision tree complexity. For example, it will imply a tight relation between decision tree complexity and square of randomized decision tree complexity and a tight relation between zero-error randomized decision tree complexity and square of fractional block sensitivity, resolving an open question raised by Aaronson [1]. In this work, we investigate the tightness of the existing cubic upper bounds.

We improve the cubic upper bounds for many interesting classes of Boolean functions. We show that for graph properties and for functions with a constant number of alternations, the cubic upper bounds can be improved to quadratic. We define a class of Boolean functions, which we call the zebra functions, that comprises Boolean functions where each monotone path from 0^n to 1^n has an equal number of alternations. This class contains the symmetric and monotone functions as its subclasses. We show that for any zebra function, decision tree complexity is at most the square of block sensitivity, and certificate complexity is at most the square of degree.

Finally, we show using a lifting theorem of communication complexity by Göös, Pitassi and Watson [21] that the task of proving an improved upper bound on the decision tree complexity for all functions is in a sense equivalent to the potentially easier task of proving a similar upper bound on communication complexity for each bi-partition of the input variables, for all functions. In particular, this implies that to bound the decision tree complexity it suffices to bound smaller measures like parity decision tree complexity, subcube decision tree complexity and decision tree rank, that are defined in terms of models that can be efficiently simulated by communication protocols.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees

Keywords and phrases Query complexity, Graph Property, Boolean functions

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.27

Related Version *Full Version*: `https://arxiv.org/pdf/2209.08042.pdf`

Funding *Rahul Chugh*: This work was done when the author was affiliated to IIT Kharagpur. Parts of this work have appeared in the B.Tech project report of the author submitted to the Department of Computer Science and Engineering, IIT Kharagpur.
Supartha Podder: Supported by US Department of Energy (grant no DE-SC0023179) and US National Science Foundation (award no 1954311).
Swagato Sanyal: Supported by an ISIRD grant by Sponsored Research and Industrial Consultancy, IIT Kharagpur.

Acknowledgements We thank anonymous referees for many helpful feedback.

© Rahul Chugh, Supartha Podder, and Swagato Sanyal;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 27; pp. 27:1–27:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The objective of computational complexity theory is to determine the amount of resources needed to solve various computational problems. A central goal of computational complexity theory is to prove limitations of general computational models meant to capture arbitrary computations that can be realized physically. Examples of such models are Turing machines and Boolean circuits. However, proving unconditional negative results for such general models seems far beyond the current reach of researchers. Hence, a popular theme of research has been to analyze more restricted models of computation, whose examples include Boolean circuits of bounded depth, communication complexity and decision trees. Such pursuits have borne fruits, and may serve as a stepping stone to build up towards a more complete understanding of general models.

This paper deals with decision tree. Decision tree is amongst the simplest models of computation. Unlike the more general models, it is often within the reach of researchers to determine the complexity of various interesting tasks in this model. A rich body of work has emerged centering the study of decision tree complexity, its variants and their connections with many other measures of complexity [12, 25]. This pursuit has resulted in a large number of measures having been defined and related to decision tree complexity. Interestingly, decision tree complexity and all of these measures have been shown to be polynomially related to one another. We refer the readers to the survey of Buhrman and de Wolf [12] in this regard. Attention has focussed on determining the exact polynomial dependence between various measures. Table 1 in [2] records the current status of our knowledge of these relations.

We now informally define the measures relevant to this work. The formal definitions may be found in the full version of this paper [15] (See Appendix A). Let f be a Boolean function and x be a generic input string. A decision tree that computes f is an algorithm that queries various input bits of an input string x, possibly adaptively, to compute $f(x)$. The complexity of a decision tree $\mathcal{A}$ that computes f, is the maximum number of input bits that $\mathcal{A}$ queries on any x (possibly adaptively) to compute $f(x)$. The *deterministic decision tree complexity* of f, denoted by $\mathrm{D}(f)$, is the minimum complexity of any decision tree that computes f. A subset $S \subseteq [n]$ of indices is called a *sensitive block* for x if flipping all bits in x with indices in S flips the value of f. The sensitivity of f on x, denoted by $\mathrm{s}(f, x)$, is the maximum number of disjoint sensitive blocks of cardinality 1 in x. The sensitivity of f, denoted by $\mathrm{s}(f)$, is defined as $\max_x \mathrm{s}(f, x)$. The block sensitivity of f on x, denoted by $\mathrm{bs}(f, x)$, is the maximum number of disjoint sensitive blocks in x. For $b \in \{0, 1\}$, the b-block sensitivity of f, denoted by $\mathrm{bs}^{(b)}(f)$, is defined to be $\max_{x \in f^{-1}(b)} \mathrm{bs}(f, x)$. The block sensitivity $\mathrm{bs}(f)$ of f is $\max\{\mathrm{bs}^{(0)}(f), \mathrm{bs}^{(1)}(f)\}$. The quantity $\mathrm{bs}(f, x)$ can be formulated as the value of an integer linear program. The value of its fractional relaxation is the fractional block sensitivity $\mathrm{fbs}(f, x)$ on x. The fractional block sensitivity $\mathrm{fbs}(x)$ is defined to be $\max_x \mathrm{fbs}(f, x)$. The exact degree $\deg(f)$ of f is the degree of the unique multi-linear polynomial P_f that computes f. Let $f(x) = b$. Then the b-certificate complexity of f on x, denoted by $\mathrm{C}_b(f, x)$, is the minimum co-dimension of a subcube $\mathcal{C}$ that contains x, such that $f(y) = b$ for every $y \in \mathcal{C}$. The *certificate complexity* (resp. *minimum certificate complexity*) of f, denoted by $\mathrm{C}(f)$ (resp. $\mathrm{C}_{\min}(f)$), is $\max_x\{C_{f(x)}(f, x)\}$ (resp. $\min_x\{C_{f(x)}(f, x)\}$). We define $\mathrm{C}^*_{\min}(f)$ to be $\max_{\mathcal{C}} \mathrm{C}_{\min}(f\mid_{\mathcal{C}})$, where $f\mid_{\mathcal{C}}$ denotes the restriction of f to $\mathcal{C}$, and the maximum is over all subcubes of the domain $\{0, 1\}^n$. It is known (see Table 1 in [2]; also follows from Proposition 29 parts (2), (3), (4) and (5)) that for all Boolean functions f, $\mathrm{D}(f) = O(\mathrm{bs}(f)^3)$ and $\mathrm{D}(f) = O(\deg(f)^3)$. It is also known that there exist Boolean functions g, h such that $\mathrm{D}(g) = \Omega(\mathrm{bs}(g)^2)$ and $\mathrm{D}(h) = \Omega(\deg(h)^2)$. In light of the above, the following is a natural question.

▶ **Question 1.** *Are the following upper bounds asymptotically tight?*

- $\mathrm{D}(f) = O(\mathrm{bs}(f)^3)$ *and*
- $\mathrm{D}(f) = O(\deg(f)^3)$

Question 1 forms the center of our study, with a special focus on the possibility of a negative answer. If the first bound can be improved to $\mathrm{D}(f) = O(\mathrm{bs}(f)^2)$ to match the widest separation known, then it would resolve several open questions in decision tree complexity. Most notably, it would imply the following tight[1] relations: $\mathrm{D}(f) = O(\mathrm{R}(f)^2)$ and quantum decision tree complexity is upper bounded by the square of fractional block sensitivity i.e., $\mathrm{Q}(f) = \widetilde{O}(\mathrm{fbs}(f)^2)$. It will also imply the tight relation: $\mathrm{R}_0(f) = O(\mathrm{fbs}(f)^2)$, resolving an open question by Aaronson [1]. Similarly, if $\mathrm{D}(f) = O(\deg(f)^2)$ is true, it will also improve several relations among measures in decision tree complexity e.g., the current best upper bound of $\mathrm{D}(f) = O(\mathrm{s}(f)^6)$ would improve to $\mathrm{D}(f) = O(\mathrm{s}(f)^4)$, and we will achieve the tight relation $\mathrm{D}(f) = O(\mathrm{Q}_E(f)^2)$, where Q_E is the exact quantum decision tree complexity.

1.1 Our results

We now give an overview of our results. We divide our results into two parts.

1.1.1 Improved upper bounds for classes of functions

Our first category of results studies Question 1 for various important classes of Boolean functions.

1.1.1.1 Graph properties

A graph property is a Boolean function where the input bit string is the adjacency matrix of an undirected graph, which is invariant under permutations of the vertices. The input to a graph property of graphs with n vertices is a string in $\{0,1\}^{\binom{n}{2}}$. Each input bit is indexed by a (unordered) pair of vertices. The value of the bit indicates whether or not there is an edge between those vertices. Decision tree complexity of graph properties has seen a long and rich line of research [42, 27, 43, 10, 44, 54, 28, 24, 13, 14, 8, 6]. We provide strong negative answers to Question 1 (2) for graph properties.

▶ **Theorem 1.** *For any graph property $\mathcal{P}$ of graphs with n vertices,* $\mathrm{D}(\mathcal{P}) = O(\deg(\mathcal{P}))^2$.

Theorem 1 follows immediately from the following lemma and the observation that $\mathrm{D}(\mathcal{P}) \leq \binom{n}{2}$.

▶ **Lemma 2.** *For any non-trivial graph property $\mathcal{P}$ on any graph $G = (V, E)$ with $|V| = n$,* $\deg(\mathcal{P}) = \Omega(n)$.

A similar result for graph properties, answering Question 1 (1), is known for sensitivity by Turán [51].

▶ **Lemma 3** (Turán [51])**.** *For any non-trivial graph property $\mathcal{P}$ on any graph $G = (V, E)$ with $|V| = n$,* $\mathrm{s}(\mathcal{P}) = \Omega(n)$.

▶ Remark. Our proof of Lemma 2 also reproves Lemma 3. However, we do not write that in the statement of Lemma 2.

[1] Some of the results will be tight up to poly-logarithmic factors.

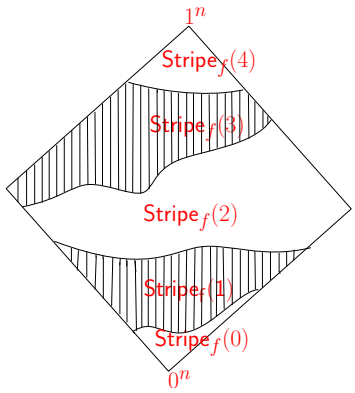


Figure 1 A zebra function with 5 stripes.

Note that, both of Lemma 2 and Lemma 3 are asymptotically tight. Their tightness is witnessed by graph properties *SINK* [43] and *SCORPION GRAPH* [10]. For both of these graph properties, $\mathrm{D}(\cdot) = O(n)$; thus $\mathrm{bs}(\cdot) = O(n)$ and $\deg(\cdot) = O(n)$. For the special case of non-trivial monotone graph properties, the bound in Lemma 2 can be improved to $\Omega(n^2)$ [18, 42]. As a result, a tight $\Omega(n)$ bound can be achieved for quantum query complexity of monotone graph properties [3]. It is also known that for every partial[2] graph properties (in the adjacency matrix model) randomized and quantum decision tree complexity are polynomially related [8].

1.1.1.2 Zebra functions

We say that a function is a *zebra function* if in each monotone path in the Boolean hypercube from 0^n to 1^n, the function value changes equal number of times. The class of zebra functions contains symmetric and monotone Boolean functions as two important sub-classes. However, there are interesting zebra functions that are neither monotone nor symmetric. One example is the *Kushilevitz's function* (Footnote 1 on page 560 of [39]), for which the value of the function is determined by the Hamming weight of the input, unless the Hamming weight is 3 [53]. Another set of examples is functions expressible as $g(\lceil \ell(x_1, \ldots, x_n) \rceil)$, where $g : \mathbb{Z} \to \{0, 1\}$ is arbitrary, and $\ell(x_1, \ldots, x_n) := \ell_0 + \sum_{i=1}^{n} \ell_i x_i$ is an affine function where each coefficient ℓ_i lies in the interval $(0, 1)$. Visually, a zebra function induces a partition of the Boolean hypercube into monochromatic "stripes" (see Figure 1). See Section 4.1 for a proof of the aforementioned stripe structure, and other properties of the zebra functions. The value of the function is constant in each stripe, and different in adjacent stripes. We are optimistic that zebra functions will turn out to be useful in future research in Boolean function complexity.

We prove the following theorem.

▶ **Theorem 4.** *For every zebra function f,*
1. $\mathrm{D}(f) = O(\mathrm{bs}(f)^2)$ *and*
2. $\mathrm{C}(f) = O(\deg(f)^2)$.

Theorem 4 (1) gives a negative answer to Question 1 (1) for zebra functions. Theorem 4 (1) is tight, as witnessed by the monotone function $\mathsf{TRIBES}_{\sqrt{n}\times\sqrt{n}}$ (see Definition A.2 in [15]). As for Theorem 4 (2), it is known that $\mathrm{C}(\cdot) = O(\mathrm{D}(\cdot))$ and $\mathrm{D}(\cdot) = O(\mathrm{C}(\cdot)^2)$ (see Proposition 29

[2] The property is not defined for all graphs.

(1) and (2)). Theorem 4 (2) thus falls short of giving a complete answer to Question 1 (2) for zebra functions. We do not know if there exists a zebra function f for which $\mathrm{C}(f)$ is polynomially larger than $\deg(f)$. If $\mathrm{C}(\cdot) = O(\deg(\cdot))$ holds for all zebra functions, then it follows from Proposition 29 (5) that $\mathrm{D}(\cdot) = O(\deg(\cdot)^2)$ holds for all zebra functions. We leave the question of whether $\mathrm{D}(f) = O(\deg(f))^2$ is true for zebra functions f as an open question.

1.1.1.3 Functions with bounded alternation number

The alternation number $\mathrm{alt}(f)$ of a Boolean function f is the maximum number of times the function value changes in any monotone path of the Boolean hypercube from 0^n to 1^n. Monotone functions are exactly those functions that are 0 on 0^n and have alternation number 1. Alternation number has received attention in the past as a measure of the extent of non-monotonicity of a Boolean function. Markov [32] related it to the number of not-gates that any Boolean circuit realizing a Boolean function must use, which is yet another measure of non-monotonicity. Alternation number has also been studied in the context of various restricted circuit models [45, 47, 35, 36], learning classes of circuits with bounded non-monotonicity [11] and cryptography [23]. Lin and Zhang [31] proved the *Sensitivity Theorem*[3] and the *Log-Rank Conjecture*, two important conjectures in complexity theory, for the special case of functions with constant alternation number. Krishnamoorthy and Sarma [17] has studied relations of alternation number with various combinatorial, algebraic and analytic complexity measures of Boolean functions.

It follows from Proposition 29 (4) and the fact that for monotone functions the measures certificate complexity and block sensitivity are equal [12] that for any monotone function f, $\mathrm{D}(f) = O(\mathrm{bs}(f)^2)$. The following theorem generalizes the above result for the class of functions with bounded alternation. In addition, it also proves a quadratic upper bound on $\mathrm{D}(f)$ in terms of $\deg(f)$ for functions with constant alternation number. In particular, it gives a negative answer to Question 1 (1) and Question 1 (2) for functions with constant alternation number.

▶ **Theorem 5.** *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$ we have*[4]
1. $\mathrm{D}(f) = O(\mathrm{alt}(f) \cdot \mathrm{s}(f) \cdot \mathrm{bs}(f))$,
2. $\mathrm{D}(f) = O(\mathrm{alt}(f) \cdot \deg(f)^2)$,

In particular, if $\mathrm{alt}(f) = O(1)$, then we get $\mathrm{D}(f) = O(\mathrm{bs}(f)^2)$ (see Proposition 29 (1)) and $\mathrm{D}(f) = O(\deg(f)^2)$.

A similar result of $\mathrm{D}(f) = O(\mathrm{alt}(f) \cdot \mathrm{s}(f) \cdot \deg(f))$, for every function f, follows from the work of Lin and Zhang [31].

To prove Theorem 5 we need some results, that we state next.

▶ **Lemma 6.** *For any Boolean function $f : \{0,1\}^n \to \{0,1\}$, $\mathrm{D}(f) = O(\mathrm{C}^*_{\min}(f) \cdot \mathrm{bs}(f))$.*

We prove Lemma 6 in Section 5. A similar bound is known for $\deg(f)$, due to Tsang [49].

▶ **Lemma 7** (Tsang [49])**.** *For any Boolean function $f : \{0,1\}^n \to \{0,1\}$, $\mathrm{D}(f) = O(\mathrm{C}^*_{\min}(f) \cdot \deg(f))$.*

Lin and Zhang [31] showed the following upper bounds on $\mathrm{C}^*_{\min}(\cdot)$.

[3] Recently, the Sensitivity Conjecture was unconditionally proved by the seminal work of Huang [26].
[4] As stated in the preceding section, $\mathsf{TRIBES}_{\sqrt{n}\times\sqrt{n}}$ witnesses the tightness of Theorem 5.

▶ **Lemma 8** (Lin and Zhang [31]). *For any Boolean function* $f : \{0,1\}^n \to \{0,1\}$,
1. $\mathrm{C}^*_{\min}(f) \leq \mathrm{alt}(f) \cdot \mathrm{s}(f)$,
2. $\mathrm{C}^*_{\min}(f) \leq \mathrm{alt}(f) \cdot \deg(f)$.

For the sake of completeness, we reproduce the proofs of Lemma 7 and Lemma 8 in the Appendix B. Our proofs deviate slightly from the original ones.

Theorem 5 (1) follows immediately from Lemma 6 and Lemma 8 (1). Theorem 5 (2) follows immediately from Lemma 7 and Lemma 8 (2).

1.1.2 Decision trees and communication protocols

In our second category of results, we reduce the task of proving improved upper bounds on $\mathrm{D}(\cdot)$ for the class of all functions to potentially easier tasks of bounding smaller measures.

Consider a bi-partition $(x^{(1)}, x^{(2)})$ of the bits of x into two parts consisting of n_1 and n_2 bits respectively. In the two-party communication model introduced by Yao [4] there are two communicating parties Alice and Bob, holding $x^{(1)}$ and $x^{(2)}$ respectively. The parties wish to compute $f(x)$ by communicating bits. The objective is to accomplish this by communicating as few bits as possible. The *communication complexity* of f with respect to the bi-partition $(x^{(1)}, x^{(2)})$, denoted by $\mathsf{CC}(f)$, is the number of bits that need to be exchanged in the worst case by any protocol that computes f.[5] See Section 2 for more details.

Suppose f has a shallow decision tree T of depth d. Then there also exists an efficient communication protocol that computes f, in which the parties simulate T . Each query that T makes can be answered by 1 bit of communication. Thus, the protocol exchanges d bits in the worst case. This gives us that $\mathsf{CC}(f) \leq \mathrm{D}(f)$.

We make the following observation.

▶ **Observation 9.**
1. *If* $\mathrm{D}(f) = O(\mathrm{bs}(f)^2)$, *then* $\mathsf{CC}(f) = O(bs(f)^2)$, *and*
2. *if* $\mathrm{D}(f) = O(\deg(f)^2)$, *then* $\mathsf{CC}(f) = O(\deg(f)^2)$.

The question we ask is whether a converse of Observation 9 holds. We show that a "approximate converse" of Observation 9 holds, up to a factor of $\mathsf{polylog}(n)$, when the antecedents and consequents are universally quantified.

▶ **Theorem 10.**
1. *If for all Boolean functions* f *and all bi-partitions of the input bits* $\mathsf{CC}(f) = O(\deg(f)^2 \cdot \mathsf{polylog}(n))$ *then for all* f, $\mathrm{D}(f) = O(\deg(f)^2 \cdot \mathsf{polylog}(n))$.
2. *If for all Boolean functions* f *and all bi-partitions of the input bits* $\mathsf{CC}(f) = O(\mathrm{bs}(f)^2 \cdot \mathsf{polylog}(n))$ *then for all* f $\mathrm{D}(f) = O(\mathrm{fbs}(f)^2 \cdot \mathsf{polylog}(n))$.

Theorem 10 opens up the possibility of deriving improved upper bounds on $\mathrm{D}(\cdot)$ by designing cheap communication protocols.[6] For instance, to prove Aaronson's conjecture (up to $\mathsf{polylog}(n)$) that the randomized query complexity is at most the fractional block sensitivity [1] it is sufficient to show that the communication complexity is at most square of the block sensitivity times $\mathsf{polylog}(n)$. Thus, our task is reduced that of bounding $\mathsf{CC}(\cdot)$. The latter is a potentially easier task; there are Boolean functions f for which

[5] The dependence of $\mathsf{CC}(\cdot)$ on the bi-partition is suppressed in the notation, and will be clear from the context.

[6] In fact, it follows from our proof that it is sufficient to design cheap protocols only for functions of the form $f \circ \mathsf{IND}_{c \log n + n^c}$ for some constant $c > 0$, where IND_{m+2^m} is the indexing function on $m + 2^m$ bits (see Definition A.3 in [15]).

$\mathsf{CC}(f) << \mathrm{D}(f)$. For example, for the majority function MAJ_n on n bits (see Definition A.4 in [15]), $\mathsf{CC}(\mathsf{MAJ}_n) \leq \lceil \log n \rceil$ for any bi-partition of its input bits, but $\mathrm{D}(\mathsf{MAJ}_n) = n$. Furthermore, there are upper bounds on $\mathsf{CC}(f)$ by measures that are bounded above by $\mathrm{D}(f)$, and are far lower than $\mathrm{D}(f)$ for many f. One such measure is the *decision tree rank* of f, denoted by $\mathrm{rank}(f)$ (See Definition A.7 in [15]).

Decision tree rank was introduced by Ehrenfeucht and Haussler [19] in the context of learning (also see [5]). More recently, Dahiya and Mahajan studied decision tree rank in the context of complexity measures of Boolean functions [16].

Decision tree rank is defined in terms of decision trees and is bounded above by $\mathrm{D}(\cdot)$. Furthermore, it can be arbitrarily smaller than $\mathrm{D}(\cdot)$. For example, for the AND function on n bits, $\mathrm{D}(\cdot) = n$ and $\mathrm{rank}(\cdot) = 1$.

To relate $\mathsf{CC}(\cdot)$ to $\mathrm{rank}(\cdot)$, we consider *subcube decision tree complexity* $\mathrm{D}^{\mathrm{sc}}(\cdot)$, (see Definition A.9 in [15]), which is the least depth of a decision tree that computes f by querying membership of the input string in various subcubes at its internal nodes. We show (see Proposition 20) that $\mathrm{rank}(\cdot)$ is an upper bound on $\mathsf{CC}(\cdot)$ up to a factor of $\log n$. We prove the proposition by showing that rank is characterized by (up to a factor of $\log n$) the *subcube decision tree complexity*, (see Definition A.9 in [15]), which is the least depth of a decision tree that computes f by querying membership of the input bit-string in various subcubes at its internal nodes. Such a tree is efficiently simulable by a communication protocol. Independently of us, Dahiya and Mahajan [16] also defined subcube decision trees (they called them CONJ-trees, also see [9]) and proved the above characterization in their work.

There are powerful decision tree models that can be efficiently simulated by communication protocols; one such model is the subcube decision tree mentioned before. Another well-known model is the *parity decision tree* (see Definition A.8 in [15]), which is a decision tree that computes f by querying the parity of various subsets of bits of the input string at its internal nodes [34, 50, 46]. A parity decision tree can be efficiently simulated by a communication protocol. Each parity query can be answered by an exchange of two bits. Thus $\mathsf{CC}(\cdot) = O(\mathsf{PDT}(\cdot))$. Theorem 10 implies that bounds on the parity decision tree complexity (denoted by $\mathsf{PDT}(\cdot)$) of the class of all Boolean functions translate to comparable bounds on the decision tree complexity of all Boolean functions.

The following theorem is a direct consequence of Theorem 10, Proposition 20, and the preceding discussion.

▶ **Theorem 11.** *Let* $\mathcal{M} \in \{\mathrm{rank}, \mathsf{PDT}\}$. *Then*

1. *If for all Boolean functions* f, $\mathcal{M}(f) = O(\deg(f)^2 \cdot \mathsf{polylog}(n))$ *then for all* f, $\mathrm{D}(f) = O(\deg(f)^2 \cdot \mathsf{polylog}(n))$.
2. *If for all Boolean functions* f, $\mathcal{M}(f) = O(\mathrm{bs}(f)^2 \cdot \mathsf{polylog}(n))$ *then for all* f, $\mathrm{D}(f) = O(\mathrm{fbs}(f)^2 \cdot \mathsf{polylog}(n))$.

Theorem 10 is proved by a lifting theorem for communication complexity of composed functions where the inner function is the indexing function on $\Theta(\log n)$ bits (see Section 1.2 and Section 6). The arity of the indexing gadget is the source of the $\mathsf{polylog}(n)$ factors in Theorem 10. A lifting theorem with a gadget of constant size is a major open problem in communication complexity, with many interesting consequences. It is apparent from our proof of Theorem 10 that such a lifting theorem will remove the polylogarithmic factors. Our study thus strengthens the case for a lifting theorem with constant sized inner function.

1.2 Proof Techniques

In this section we discuss at a high level the key ideas of our proofs.

Graph properties

To prove Lemma 2, we assume, by complementing the property $\mathcal{P}$ if necessary, that the empty graph is not in the property. Then we split the proof into two parts based on the smallest number of edges m of any graph $G = (V, E)$ in the property.

If $m > n/4$, then we consider the restriction of $\mathcal{P}$ where each $x_{\{u,v\}}$ such that $\{u, v\} \notin E$ is set to 0. We observe that the restricted function is the logical AND of the unset variables, and hence has block sensitivity and degree both equal to $m = \Omega(n)$.

On the other hand, if $m \leq n/4$, then we show that by carefully restricting and identifying variables, it is possible to turn $\mathcal{P}$ into a non-trivial symmetric function on $\Omega(n)$ variables which, by known results, has both degree and block sensitivity $\Omega(n)$ (see Proposition 29 (9)).[7]. In this part of the proof, we crucially use the invariance of $\mathcal{P}$ under permutation of vertices.

Zebra functions and functions with bounded alternation

In the proofs of Theorem 4 (1) as well as Theorem 5, the Lemma 6 and Lemma 7 play a central role. Lemma 6 is proved by a natural adaptation of the proofs Proposition 29 (3) and (4) [12, 33] and follows a similar argument of the proof of Lemma 7 [49]. We provide a proof of Lemma 6 in Section 5. In light of Lemma 6, the proofs of the aforementioned results boil down to bounding the maximin certificate complexity of the functions in the respective classes. To prove Theorem 4 (2), we bound the number of bits that need to be revealed to certify the stripe a specific input belongs to. To do that, we consider the monotone and anti-monotone functions obtained by treating the boundary of a specific stripe as a threshold, bound the certificate complexities of these functions, and show how those certificates can be put together to certify membership of a certain input in a specific stripe.

Communication complexity

The proof of Theorem 10 is based on a lifting theorem for deterministic communication complexity (see Theorem 21), which essentially asserts that there for every function g, the deterministic communication complexity of $g \circ \mathsf{IND}_{c \log n + n^c}$ is asymptotically the same as $\log n \cdot \mathrm{D}(g)$, where IND_{m+2^m} is the indexing function on $m + 2^m$ bits (see Definition A.3 in [15]). The hypotheses of the two parts of Theorem 10, coupled with the lifting theorem, immediately implies upper bounds on $\mathrm{D}(g)$ in terms of the block sensitivity and degree of $g \circ \mathsf{IND}_{c \log n + n^c}$. We finish the proof by bounding the block sensitivity and degree of $g \circ \mathsf{IND}_{c \log n + n^c}$ in terms of the fractional block sensitivity and degree of g respectively.

1.3 Organization of the paper

In Section 2 we lay down some notations that are used in this paper, and give some definitions. In Section 3 we define graph properties, set some notations and prove Theorem 1. In Section 4 we formally define zebra functions and related concepts, state some basic properties of zebra

[7] Even the sensitivity of a non-constant symmetric function is $\Omega(n)$ (see Proposition 29 (9)). Thus this reproves Turán's result Lemma 3.

functions, and prove Theorem 4. In Section 5 we prove the result about functions with bounded alternation number (Theorem 5). In Section 6 we prove our main result (Theorem 10) pertaining to connections of Question 1 to communication complexity and other decision tree models. The appendices contain alternative proofs of known results. We will be referring to our full paper [15] for some definitions.

2 Definitions and Preliminaries

For any real number $t > 0$, $\log t$ stands for the logarithm of t to the base 2. We denote the set of all integers by $\mathbf{Z}$. For a natural number N, $[N]$ denotes the set $\{1, \ldots, N\}$. Let $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in \{0,1\}^n$. We say that $x \leq y$ (resp. $x \geq y$) if for each $i \in [n]$, $x_i \leq y_i$ (resp. $x_i \geq y_i$). We say that $x < y$ (resp. $x > y$) if $x \leq y$ (resp. $x \geq y$) and $x \neq y$. $|x|$ denotes the *Hamming weight* of x defined as $\sum_{i=1}^{n} x_i$. A 1-bit (resp. 0-bit) of x is an index $i \in [n]$ such that $x_i = 1$ (resp. $x_i = 0$). For $S \subseteq [n]$, let $\{0,1\}^S$ denote the set of all binary strings whose indices correspond to elements of S, i.e., if $S = \{i_1, \ldots, i_k\}$ then $\{0,1\}^S = \{(b_{i_1}, \ldots, b_{i_k}) \mid b_{i_j} \in \{0,1\} \text{ for } j = 1, \ldots, k\}$. For a set of indices $B \in [n]$, we denote the string obtained from x by negating the bits in the locations with indices in B by $x^{\oplus B}$. If $B = \{i\}$ is singleton, we abuse notation and write $x^{\oplus B}$ as $x^{\oplus i}$. for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a bit $b \in \{0,1\}$, $f^{-1}(b) := \{x \in \{0,1\}^n \mid f(x) = b\}$.

▶ **Definition 12** (Monotone increasing path or monotone path, alternation number of monotone paths and functions). *Let $p, q \in \{0,1\}^n$ such that $p \leq q$. A* monotone increasing path *from p to q in the Boolean hypercube $\{0,1\}^n$ is a sequence $(x^{(1)}, \ldots, x^{(k)})$ such that*

- *Each element $x^{(i)}$ of the sequence is a bit string $(x_1^{(i)}, \ldots, x_n^{(i)}) \in \{0,1\}^n$, for $i = 1, \ldots, k$,*
- *$p = x^{(1)}$ and $q = x^{(k)}$,*
- *for each $i = 1, \ldots, k-1$, there exists a $j \in [n]$ such that*
  - *$x_j^{(i)} = 0$ and $x_j^{(i+1)} = 1$,*
  - *$x_\ell^{(i)} = x_\ell^{(i+1)}$ for each $\ell \in [n] \setminus \{j\}$.*

We refer to such a path simply as a monotone path *for convenience.*

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. The alternation number *of a monotone path $P = (x^{(1)}, \ldots, x^{(k)})$ with respect to f, denoted by $\mathrm{alt}_f(P)$, is the number of times f changes value on the path. Formally,*

$$\mathrm{alt}_f(P) = |\{i \in [k-1] \mid f(x^{(i)}) \neq f(x^{(i+1)})\}|.$$

We often drop the subscript f from the notation and denote alternation number of P simply by $\mathrm{alt}(P)$ when the function is clear from the context.

The alternation number of f, denoted by $\mathrm{alt}(f)$, is defined to be the maximum alternation number of any monotone path from 0^n to 1^n with respect to f.

▶ **Observation 13.** *Let $x, y \in \{0,1\}^n$ and $x \leq y$. Then there exists a monotone path from x to y.*

▶ **Observation 14.** *The alternation number of a function f is at least the alternation number of any restriction of f. That is, for every integer $k \geq 0$, if $\mathrm{alt}(f) \leq k$ then for every subcube C, $\mathrm{alt}(f\mid_C) \leq k$.*

3 Graph Properties

In this section, we first define graph properties and lay down few notations. We then proceed to prove Lemma 2, which immediately implies Theorem 1. The proof of Lemma 2 proceeds along the lines of the proof of Lemma 3 by Turán [51].

Let $\mathcal{G}_n$ be the set of all simple undirected graphs on n-vertices. A graph property is a function $\mathcal{P} : \mathcal{G}_n \to \{0,1\}$ which is invariant under permutation of vertices. In other words, if two graphs are isomorphic, then either both belong to the property, or both do not.

Example of a graph property is one that maps connected graphs to 1 and disconnected graphs to 0. But, the Boolean function that maps exactly those graphs that have an edge between vertices 1 and 2 to 1, is not a graph property, as it is not invariant under relabling of vertices.

A non-constant graph property is also called *non-trivial*. A graph $G \in \mathcal{G}_n$ is often identified with a string $x_G \in \{0,1\}^{\binom{n}{2}}$. Each location of x_G corresponds to a distinct unordered pair of distinct vertices of G. $(x_G)_{\{u,v\}} := 1$ if there is an edge in G between vertices u and v, and $(x_G)_{\{u,v\}} := 0$ otherwise. We use $\mathcal{P}(G)$ and $\mathcal{P}(x_G)$ interchangeably.

Proof of Lemma 2. Wlog. we assume that the empty graph is not in $\mathcal{P}$. Let $m > 0$ be the least number of edges in any graph in $\mathcal{P}$. Let $G = (V, E) \in \mathcal{P}$ have m edges. We consider the following cases.

Case 1 $m > \frac{n}{4}$: Restrict $\mathcal{P}$ by setting $(x_G)_{\{u,v\}}$ to 0 for each $\{u, v\} \notin E$. By the minimality of G, the restriction is the AND function on the m variables corresponding to the edges in G. Thus the block sensitivity and degree of the restriction are both $m > n/4$. Since degree and block sensitivity do not increase under restriction (Proposition 29 (7)), therefore $\mathrm{bs}(\mathcal{P}) > n/4$ and $\deg(\mathcal{P}) > n/4$.

Case 2 $m \leq \frac{n}{4}$: In the graph G, there are at least $\frac{n}{2}$ vertices that have degree zero. Let the vertices with non-zero degrees be $v_1, v_2, \cdots v_k$, where $k \leq \frac{n}{2}$. Let the remaining vertices be $v_{k+1}, \cdots v_n$. Without loss of generality let v_k be connected to $v_1, \cdots, v_d$. Let G' be the graph obtained from G by removing all edges incident on v_k. Since G' has less than m edges, $G' \notin \mathcal{P}$. Now for each $j \in \{k, k+1, \cdots, n\}$, consider the graph G_j obtained by adding the edges $\{j, v_1\}, \{j, v_2), \cdots \{j, v_d\}$ to G'. All the G_js formed this way are isomorphic to G, and are thus all in $\mathcal{P}$. Also note that G_k is the same as G.

Now consider a function $f : \{0,1\}^{n-k+1} \to \{0,1\}$ defined as follows.
Let $x = (x_k, x_{k+1}, \ldots, x_n) \in \{0,1\}^{n-k+1}$. We will now define a graph G_x on n vertices, and then define $f(x) := \mathcal{P}(G_x)$. G_x contains exactly the following edges.

- All edges of G'.
- Edges $\{v_i, v_1\}, \ldots, \{v_i, v_d\}$ for each $i \in \{k, k+1, \ldots, n\}$ such that $x_i = 1$.

First, note that $f(0^{n-k+1}) = 0$ (since $G_{0^{n-k+1}} = G' \notin \mathcal{P}$) and for any string x with $|x| = 1$, $f(x) = 1$ (since in this case G_x is the same as G_j for some $j \in \{k, \ldots, n\}$, and is hence in $\mathcal{P}$). Thus, f is a non-constant function.

Next, note that for any $x, x' \in \{0,1\}^{n-k+1}$ such that $|x| = |x'|$, G_x and $G_{x'}$ are isomorphic, and hence $f(x) = \mathcal{P}(G_x) = \mathcal{P}(G_{x'}) = f(x')$. Thus, f is a symmetric function.

Since any non-constant symmetric function has linear degree and block sensitivity (Proposition 29 (8)), therefore $\deg(f)$ and $\mathrm{bs}(f)$ are both $\Omega(n-k+1) = \Omega(n)$ (since $k \leq n/2$).

Finally, f is obtained from $\mathcal{P}$ by restriction and identification of variables, as follows.

- Every variable corresponding to a pair $\{v_i, v_j\}$ such that $1 \leq i < j \leq k-1$ is set to 0 if $\{i, j\}$ is not an edge of G' and set to 1 otherwise.
- For each $i \in \{k, k+1, \ldots, n\}$, the variables corresponding to pairs $\{v_i, v_1\}, \ldots, \{v_i, v_d\}$ are identified.
- All other variables are set to 0.

Since degree and block sensitivity do not increase under restriction and identification of variables (Proposition 29 parts (7) and (8)), therefore, $\deg(\mathcal{P})$ and $\mathrm{bs}(\mathcal{P})$ are both $\Omega(n)$. ◀

4 Zebra functions

In this section, we first formally define zebra functions and some related concepts, and state some basic facts about zebra functions. We then proceed to prove Theorem 4.

4.1 Definitions and basic concepts

▶ **Definition 15** (Zebra function). *A function $f : \{0,1\}^n \to \{0,1\}$ is called a* zebra function *if all monotone paths from 0^n to 1^n have same alternation number.*

Visually, a zebra function induces a partition of the Boolean hypercube into *stripes*. In each stripe, the value of the function is constant (i.e., each stripe is monochromatic with respect to the function). Furthermore, the value of the function is different in adjacent stripes. Each monotone path from 0^n to 1^n must pass through each stripe (See Figure 1). We now build up towards defining the stripes formally.

▶ **Proposition 16.** *Let f be a zebra function on n bits and $x, y \in \{0,1\}^n$ be such that $x \leq y$. Let P_1 and P_2 be two monotone paths from x to y. Then, $\mathrm{alt}_f(P_1) = \mathrm{alt}_f(P_2)$.*

Proof. Towards a contradiction, assume that $\mathrm{alt}_f(P_1) \neq \mathrm{alt}_f(P_2)$. Consider any monotone paths Q_1 and Q_2 from 0^n to x and from y to 1^n respectively. Let R_1 (resp. R_2) be the the monotone path from 0^n to 1^n obtained by concatenating Q_1, P_1 and Q_2 (resp. Q_1, P_2 and Q_2). Then, $\mathrm{alt}_f(R_1) = \mathrm{alt}_f(Q_1)+\mathrm{alt}_f(P_1)+\mathrm{alt}_f(Q_2)$ and $\mathrm{alt}_f(R_2) = \mathrm{alt}_f(Q_1)+\mathrm{alt}_f(P_2)+\mathrm{alt}_f(Q_2)$. Thus, $\mathrm{alt}_f(R_1) \neq \mathrm{alt}_f(R_2)$, contradicting the hypothesis that f is a zebra function. ◀

In light of Proposition 16 we extend the definition of alternation number to points with respect to zebra functions.

▶ **Definition 17** (Alternation number of a point with respect to a zebra function). *Let f be a zebra function on n bits and $x \in \{0,1\}^n$. The alternation number of x with respect to f, denoted by $\mathrm{alt}_f(x)$, is defined as $\mathrm{alt}_f(P)$ for any monotone path from 0^n to x. Proposition 16 guarantees that $\mathrm{alt}_f(x)$ is well-defined.*

Now we are ready to define stripes of a zebra function. Let f be a zebra function on n bits and $\mathrm{alt}(f) = k$. For $i = 0, \ldots, k$, the i-th stripe of f, $\mathrm{STRIPE}_f(i)$, is defined to be the set $\{x \in \{0,1\}^n \mid \mathrm{alt}_f(x) = i\}$. An easy induction on i establishes that $\mathrm{STRIPE}_f(i)$ is monochromatic with respect to f. For $i = 1, \ldots, k$, we denote by $f(i)$ the value of f on any point in $\mathrm{STRIPE}_f(i)$.

$x \in \mathrm{STRIPE}_f(i)$ is called a *minimal point* of $\mathrm{STRIPE}_f(i)$ if for each point $y < x$, $y \notin \mathrm{STRIPE}_f(i)$. Similarly, $x \in \mathrm{STRIPE}_f(i)$ is called a *maximal point* of $\mathrm{STRIPE}_f(i)$ if for each point $y > x$, $y \notin \mathrm{STRIPE}_f(i)$.

The following proposition lists some useful facts about zebra functions.

▶ **Proposition 18.** *Let f be a zebra function on n bits and $\mathrm{alt}(f) = k$. Then the following hold.*

1. *For $0 \leq i < j \leq k$, $\mathrm{STRIPE}_f(i) \cap \mathrm{STRIPE}_f(j) = \emptyset$. Also, $\cup_{i=1}^{k} \mathrm{STRIPE}_f(i) = \{0,1\}^n$. In other words, the stripes form a partition of the Boolean hypercube.*
2. *For $i = 0, \ldots, k-1$, $f(i) \neq f(i+1)$.*
3. *Let $i \in \{0, \ldots, k\}$ and $x \in \mathrm{STRIPE}_f(i)$. Then there exists a $y \leq x$ (resp. $y \geq x$) such that y is a minimal (resp. maximal) point of $\mathrm{STRIPE}_f(i)$.*

4. *Let $x = (x_1, \ldots, x_n)$ be a minimal (resp. maximal) point of* $\mathrm{STRIPE}_f(i)$. *Let $x_i = 1$ (resp. $x_i = 0$). Let y be a point obtained from x by changing the value of x_i and leaving other bits of x unchanged. Then $y \in \mathrm{STRIPE}_f(i-1)$ (resp. $y \in \mathrm{STRIPE}_f(i+1)$). In particular, $f(y) \neq f(x)$.*
5. *For any $j \in [n]$ and $b \in \{0,1\}$, the function $f\mid_{x_j=b}$ is a zebra function. In other words, the class of all zebra functions is closed under restrictions of variables.*

Proof.
(Part 1) Follows immediately from the definition of stripes.
(Part 2) Suppose without loss of generality that $f(0^n) = 0$. Now, one can show that $x \in \{0,1\}^n$, $f(x) = \mathrm{alt}_f(x) \mod 2$ by an easy induction on $|x|$ ($|x|+1$ is the number of points on any monotone path from 0^n to x). This completes the proof of this part.
(Part 3) Consider set $S = \{z : z \leq \text{ and } x \in \mathrm{STRIPE}_f(i)\}$. $S \neq \phi$ as $x \in S$. The proof follows by considering a point in S with minimum hamming weight.
(Part 4) Without loss of generality, let y be obtained by flipping a 1-bit of x to 0. Since x is a minimal point of $\mathrm{STRIPE}_f(i)$ we have $y \notin \mathrm{STRIPE}_f(i)$. Considering any monotone path from 0^n to x through y, we conclude that $\mathrm{alt}_f(x) = \mathrm{alt}_f(y) + 1$. the proof follows.
(Part 5) For any string $x \in \{0,1\}^{[n]\setminus\{j\}}$, define a string $e_j(x)$ to be the following string in $\{0,1\}^n$:

$$\forall k \in [n], (e_j(x))_k = \begin{cases} x_k & \text{if } k \neq j, \\ b & \text{otherwise.} \end{cases}$$

Then, for all $y \in \{0,1\}^{[n]\setminus\{j\}}$, $f\mid_{x_j=b}(y) = f(e_j(y))$. Thus the alternation number of a monotone path from $0^{[n]\setminus\{j\}}$ to $1^{[n]\setminus\{j\}}$ with respect to $f\mid_{x_j=b}$ is equal to the alternation number of some monotone path from $e_j(0^{[n]\setminus\{j\}})$ to $e_j(1^{[n]\setminus\{j\}})$ with respect to f. The proof now follows by Proposition 16. ◀

4.2 Proof of Theorem 4

We need the following lemma.

▶ **Lemma 19.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a zebra function and* $\mathrm{alt}(f) = k$. *For $i = 0, \ldots, k$, define*

$$f_i(x) = \begin{cases} 1 & \text{if } x \in \cup_{j \geq i} \mathrm{STRIPE}_f(j), \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\mathrm{C}(f_i) = O(\deg(f)^2)$.

Proof. The Lemma is trivial for $i = 0$. So, we assume that $1 \leq i \leq k$. We show that $\mathrm{C}_1(f_i) = O(\deg(f)^2)$. The argument that $\mathrm{C}_0(f_i) = O(\deg(f)^2)$ is analogous. First, note that f_i is a zebra (monotone) function with stripes $\mathrm{STRIPE}_{f_i}(0) = \cup_{j<i}\mathrm{STRIPE}_f(j)$ and $\mathrm{STRIPE}_{f_i}(1) = \cup_{j \geq i}\mathrm{STRIPE}_f(j)$. Next, fix an input x such that $f_i(x) = 1$. Thus $x \in \mathrm{STRIPE}_f(j)$ for some $j \geq i$. Let $y \leq x$ be a minimal input of $\mathrm{STRIPE}_f(i)$ (Proposition 18 (3)). y is also a minimal input of $\mathrm{STRIPE}_{f_i}(1)$. Thus $f_i(y) = 1$. Since f_i is monotone the 1-bits of y form a certificate of x with respect to f_i. By Proposition 18 (4), all the 1-bits of y are sensitive with respect to f. This proves that $\mathrm{C}_1(f_i, x) \leq s(f) \leq \deg(f)^2$ (Proposition 29 (3)). Since x is an arbitrary 1-input, the claim follows. ◀

Proof of Theorem 4. Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant zebra function.

(Part 1). By Lemma 6 it is sufficient to prove that $\mathrm{C}^*_{\min}(f) \leq bs(f)$. Let $\mathrm{alt}(f) = k \geq 1$, and let $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ be a minimal point in $\mathrm{STRIPE}_f(k)$ (Proposition 18(3)). Let t be the number of 1-bits in x. By Proposition 18(4), every 1-bit in x is sensitive. Thus, $\mathrm{bs}(f) \geq \mathrm{s}(f) \geq t$. On the other hand, every input $(y_1, \ldots, y_n)$ such that $y_i = 1$ whenever $x_i = 1$, lies in $\mathrm{STRIPE}_f(k)$, and hence $f(y) = f(x)$; thus the 1-bits of x form a certifcate of x. Hence $\mathrm{C}_{\min}(f) \leq \mathrm{C}(f, x) \leq t \leq \mathrm{bs}(f)$. Now since the class of zebra functions is closed under restriction of variables (Proposition 18(5)), and block sensitivity does not grow under restriction of variables, therefore for any restriction f' of f to a subcube, $\mathrm{C}_{\min}(f') \leq \mathrm{bs}(f') \leq \mathrm{bs}(f)$. Since, this holds for every restriction f' of f, we conclude that $\mathrm{C}^*_{\min}(f) \leq \mathrm{bs}(f)$, and the proof is complete.

(Part 2). Let $x \in \mathrm{STRIPE}_f(i)$. Let the functions f_i be as defined in Lemma 19. By Lemma 19, x has a 1-certificate c_1 of size $O(\deg(f))^2$ with respect to f_i. If $i = k$, that also certifies that $x \in \mathrm{STRIPE}_f(k)$. Since, $\mathrm{STRIPE}_f(k)$ is monochromatic with respect to f, we are through. If $i < k$, by Lemma 19 x has a 0-certificate c_0 of size $O(\deg(f))^2$ with respect to f_{i+1}. The concatenation of c_1 and c_0 certifies that $x \in \mathrm{STRIPE}_f(i)$. Since, $\mathrm{STRIPE}_f(i)$ is monochromatic with respect to f, the theorem follows. ◀

5 Functions with bounded alternation and maximin certificate complexity

In this section we prove Lemma 6.

Proof of Lemma 6. Consider the following decision tree computing f.

Algorithm 1

Input: $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.
$\mathcal{D} \leftarrow \{0,1\}^n, g \leftarrow f$.
while *g is not constant* **do**
  Let $C = \{y \in \mathcal{D} \mid y_i = a_i \forall i \in S\}$ be a certificate of g with least co-dimension.
  Query x_i for all $i \in S$. let the outcomes be $x_i = a'_i$ for all $i \in S$.
  $\mathcal{D} \leftarrow \{y \in \mathcal{D} \mid y_i = a'_i \forall i \in S\}$.
  $g \leftarrow g\,|_{\mathcal{D}}$.
end
Return value of constant function g.

Algorithm 1 is easily seen to always halt and compute f.

We will show that Algorithm 1 makes at most $\mathrm{C}^*_{\min}(f) \cdot (\mathrm{bs}^{(0)}(f) + \mathrm{bs}^{(1)}(f)) \leq 2\mathrm{C}^*_{\min}(f) \cdot \mathrm{bs}(f)$ queries.

Towards a contradiction, suppose if possible that the number of iterations is strictly more than $\mathrm{bs}^{(0)}(f) + \mathrm{bs}^{(1)}(f)$ for some input. Then Algorithm 1 queries either at least $\mathrm{bs}^{(1)}(f) + 1$ 0-certificates or at least $\mathrm{bs}^{(0)}(f) + 1$ 1-certificates. We assume without loss of generality that the algorithm queries at least $\mathrm{bs}^{(1)}(f) + 1$ 0-certificates. Consider the domain $\mathcal{D}$ right after the $\mathrm{bs}^{(1)}(f)$-th 0-certificate is queried. Since $f\,|_{\mathcal{D}}$ is not a constant function there exits an input $z \in \mathcal{D}$ such that $f(z) = 1$. Now, for each 0-certificate C queried so far, the set of

indices where z and C disagree is a sensitive block of z. Furthermore, these blocks are all disjoint, as each time the algorithm chooses a certificate that is consistent with the query outcomes so far. It follows that the algorithm has already fully queried $\mathrm{bs}(f) \geq \mathrm{bs}(f, z)$ many disjoint blocks of z, and the answers are all consistent with z. Since these bits form a 1-certificate, we have reached a contradiction to the fact the $f|_{\mathcal{D}}$ is non-constant. This completes the proof. ◀

6 Connections to communication complexity and other measures

In this section we prove Theorem 10 and Theorem 11. First, we prove that decision tree rank is equivalent up to a factor of $\log n$, to the subcube decision tree complexity. Dahiya and Mahajan [16] independently arrived at the definition of subcube decision tree complexity and the aforementioned characterization.

▶ **Proposition 20.**

1. *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$ and for any arbitrary bi-partition of the indices into two disjoint sets $S_1, S_2 \subseteq [n]$ such that $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = [n]$,*

$$\mathsf{CC}(f) \leq 2 \cdot \mathrm{D}^{\mathrm{sc}}(f) \leq 2 \cdot \mathrm{D}(f).$$

2. *For every Boolean function f, $\mathrm{rank}(f) \leq \mathrm{D}^{\mathrm{sc}}(f) \leq \mathrm{rank}(f) \cdot (\log n + 1)$.*

Proof.

(Part 1). The first inequality follows from the observation that a communication protocol can simulate a subcube decision tree of depth d by exchanging at most $2d$ bits. Each query of a subcube decision tree can be evaluated by an exchange of 2 bits, as follows. Let the subcube queried at an internal node be $C = \{x \in \{0,1\}^n \mid x_i = b_i \forall i \in S\}$. For each $i \in S$, x_i is held by one of the parties. Thus, each party may separately check if the values of all the variables x_i for $i \in S$ that they hold match with a_i, exchange the outcomes of those checks, and thus answer the subcube query of membership in C. The second inequality holds as a decision tree is a special kind of subcube decision tree where all the subcubes queried have co-dimension 1.

(Part 2). For the first inequality, we prove by induction on t that for every subcube decision tree T of depth t, there exists a decision tree T' of rank at most t that computes the same function. The base case $t = 0$ is trivial. Now, let X be a subcube decision tree of f of depth $t \geq 1$. Let a be its root, querying subcube $\{y \in \{0,1\}^n \mid y_{i_1} = a_{i_1}, \ldots, y_{i_k} = a_{i_k}\}$, and the two subtrees of a be X_1 and X_2, each of depth at most $t-1$. By inductive hypothesis, there exist two decision trees T_1 and T_2, each of rank at most $t-1$, that compute same functions as X_1 and X_2 respectively. Next, we replace a by a decision tree T'' that queries the input variables with indices $i_1, \ldots, i_k$ and decides on the outcome of the subcube query. T'' queries the variables in order, and outputs 0 if it finds a i_j such that $y_{i_j} \neq a_{i_j}$. If it finds no such i_j it outputs 1. Next, we replace each leaf of T'' by either X_1 or X_2 depending on the outcome of the subcube query in that leaf. It is easy to check that the rank of the resulting tree is at most $(t-1)+1 = t$.

Now we prove the second inequality. Let T be a decision tree with rank $\mathrm{rank}(f) = r \geq 1$, say, and depth $d \leq n$, that computes f. We will show that there is a subcube decision tree T' of depth at most $r(\log n + 1)$ that computes f.

Let v be any node of T. Let the variables queried on the unique path from the root of T to v be $x_{i_1}, \ldots, x_{i_\ell}$ and the answers to these queries be $a_{i_1}, \ldots, a_{i_\ell}$. Thus, if T is run on an input x, the computation visits node v *if and only if* x belongs to the subcube

$\{y \in \{0,1\}^n \mid y_{i_1} = a_{i_1}, \ldots, y_{i_k} = a_{i_k}\}$. Thus, a single subcube query suffices to determine whether a particular node of T is visited. Extending this idea, we may do a binary search on the unique path in T from its root to v to find out the first vertex off the path that the computation visits when T is run on x. The number of subcube queries spent is at most one more the logarithm of the depth of v; in particular, it is at most $\log n + 1$.

Now, we describe T' recursively. Let x be an input. Let v be a deepest vertex in T such that $\mathrm{rank}(v) = r$. Let P denote the path in T from its root to v. Using the idea of the preceding paragraph T' first finds out, spending at most $\log n + 1$ queries, the first vertex $u \notin P$ in T that the computation visits when T is run on x. Next, we show that $\mathrm{rank}(u) \leq r - 1$. To see this, first note that u is a child of a vertex $w \in P$. Now, if $w \neq v$, then w has a child that is in P and hence has rank r. Since $\mathrm{rank}(w) = r$, therefore the rank of the other child u of w must be strictly less than r. On the other hand, if $w = v$, then $\mathrm{rank}(u) \leq r - 1$ by the definition of v. Next, T' recursively runs the subtree of T rooted at u on x, whose rank is at most $r - 1$ by the preceding argument. The complexity of T' is readily seen to be at most $r \cdot (\log n + 1)$. ◀

Let IND_{m+2^m} denote the indexing function on $m + 2^m$ bits (see Definition A.3 in [15]). Göös, Pitassi and Watson proved the following lifting theorem by using a simulation method by Raz and Mckenzie, which essentially asserts that the communication protocol for $f \circ \mathsf{IND}_{c \log n + n^c}$ that simulates an optimal decision tree of f, serving each query by solving the corresponding copy of the indexing function, is asymptotically optimal.

▶ **Theorem 21** ([22, 41]). *There exists a constant $c > 0$ such that for every Boolean function $f : \{0,1\}^n \to \{0,1\}$,*

$$\mathsf{CC}(f \circ \mathsf{IND}_{c \log n + n^c}) = \Theta(\log n \cdot D(f)),$$

where the communication complexity is with respect to the bi-partition of the input variables where the address bits of the input string of each copy of indexing are with Alice, and the corresponding target bits are with Bob.

The following proposition gives an exact relation between the degree of composition of two functions and the degrees of the individual functions.

▶ **Proposition 22** ([48]). *Let $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$ be two Boolean functions, then* $\deg(f \circ g) = \deg(f) \cdot \deg(g)$

Now we are ready to prove Theorem 10.

Proof of Theorem 10.

(Part 1). Assume the hypothesis of the statement of the theorem and let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary Boolean function. Let c be the constant from Theorem 21, and $m = c \log n$. Consider the function $F = f \circ \mathsf{IND}_{m+2^m}$. Consider the bi-partition of the input bits of F, in which Alice gets the address bits and Bob gets the target bits of the inputs to each copy of the inner indexing function. Invoking the hypothesis of the theorem on F with respect to this bi-partition, we have that $\mathsf{CC}(F) = O(\deg(F)^2 \cdot \mathsf{polylog}(n))$. Combining this with Theorem 21 we get that $\mathrm{D}(f) = O(\mathsf{CC}(F)/m) = O(\deg(F)^2 \cdot \mathsf{polylog}(n))$.

Then using Proposition 22 and by noting that $\deg(\mathsf{IND}) = m + 1$ we get $\mathrm{D}(f) = O(\deg(f)^2 \cdot \mathsf{polylog}(n))$.

(Part 2). Assume the hypothesis of the statement of the theorem, let $f : \{0,1\}^n \to \{0,1\}$ be any arbitrary Boolean function, and let F be defined as in part 1. Invoking the hypothesis of the theorem on F with respect to the bi-partition of part 1, we have that $\mathsf{CC}(F) = O(\mathrm{bs}(F)^2 \cdot \mathsf{polylog}(n))$. Combining this with Theorem 21 we get that $\mathrm{D}(f) = O(\mathsf{CC}(F)/m) = O(\mathrm{bs}(F)^2 \cdot \mathsf{polylog}(n))$. We finish by showing that $\mathrm{bs}(F) \leq (m+1) \cdot \mathrm{fbs}(f)$, which is what the rest of this proof focusses on.

Let $\mathrm{bs}(F) = k$. We will show that $\mathrm{fbs}(f) \geq \frac{k}{m+1}$. Let $x := (x^{(1)}, x^{(2)}, \cdots, x^{(n)})$ be an input such that $\mathrm{bs}(F, x) = k$. Let $\mathcal{B} = \{B_1, B_2, \cdots, B_k\}$ be a set of k minimal disjoint sensitive blocks of x.

We claim that for $i \in [k]$, at most $(m+1)$ blocks from $\mathcal{B}$ intersect $x^{(i)}$. To establish the claim, we start by observing that to change the output of indexing function on $x^{(i)}$ it is necessary to either change an address bit of $x^{(i)}$, or the unique target bit pointed to by the address bits. It follows from the minimality of the blocks in $\mathcal{B}$ that if a B_j overlaps nontrivially with a $x^{(i)}$, then B_j must contain either an address bit of $x^{(i)}$, or the unique target bit in $x^{(i)}$ that the address bits point to. Since there are m address bits, one target bit, and the blocks in $\mathcal{B}$ are disjoint, the claim follows.

Define $y_i := \mathsf{IND}_{m+2^m}(x^{(i)})$ and define $y = (y_1, \ldots, y_n)$. Note that y is in the domain of f. For each block $B_j \in \mathcal{B}$ define $A_j = \{\ell \in [n] \mid x^{(\ell)} \cap B_j \neq \emptyset\}$. By the minimality of the blocks in $\mathcal{B}$ each A_j is a sensitive block of y with respect to f.

Consider the following assignment of weights to the sensitive (with respect to f) blocks of y. Each sensitive block A of y is assigned weight: $w_A := \frac{|\{j \in [k] | A_j = A\}|}{m+1}$. We first show that this assignment is a feasible point of the fractional block sensitivity LP. Fix an $i \in [n]$. Now, each sensitive block of y of positive weight that contains i is in fact A_j for some $j \in [k]$ such that $B_j \cap x^{(i)} \neq \emptyset$. Now, since we have shown that the number of such j's is at most $m+1$, it follows that the sum of the weights of all sensitive blocks of y containing i is at most 1. Thus the assignment is feasible for the fbs LP. Hence,

$$\begin{aligned}
\mathrm{fbs}(f) &\geq \sum_A w_A \\
&= \sum_A \frac{|\{j \in [k] \mid A_j = A\}|}{m+1} \\
&= \frac{k}{m+1}
\end{aligned}$$

where the all the sums are over all the sensitive blocks of y. ◀

We now prove Theorem 11 which is an immediate corollary of Theorem 10.

Proof of Theorem 11: Assume the hypothesis of the theorem. By Proposition 20 (1) and (2) for any Boolean function f, and for any bi-partition of f,

$$\mathsf{CC}(f) = O(\mathrm{D}^{\mathrm{sc}}(f)) = O(\mathrm{rank}(f) \cdot \mathsf{polylog}(n)).$$

It is easy to see that a communication protocol can answer each query of a parity decision tree by exchanging 2 bits. Thus, $\mathsf{CC}(f) = O(\mathsf{PDT}(f))$. The theorem follows from Theorem 10. ◀

References

1. Scott Aaronson. Quantum certificate complexity. *Journal of Computer and System Sciences*, 74(3):313–322, 2008.
2. Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1330–1342, 2021.
3. Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1330–1342, 2021.
4. C-C Yao Andrew. Some complexity questions related to distributed computing. In *Proc. 11th STOC*, pages 209–213, 1979.
5. James Aspnes, Eric Blais, Murat Demirbas, Ryan O'Donnell, Atri Rudra, and Steve Uurtamo. k+ decision trees. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 74–88. Springer, 2010.
6. Nikhil Balaji, Samir Datta, Raghav Kulkarni, and Supartha Podder. Graph properties in node-query setting: effect of breaking symmetry. *arXiv preprint*, 2015. `arXiv:1510.08267`.
7. Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001.
8. Shalev Ben-David, Andrew M Childs, András Gilyén, William Kretschmer, Supartha Podder, and Daochen Wang. Symmetries, graph properties, and quantum speedups. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 649–660. IEEE, 2020.
9. Yosi Benasher and Ilan Newman. Decision trees with boolean threshold queries. *Journal of Computer and System Sciences*, 51(3):495–502, 1995.
10. M.R. Best, P. van Emde Boas, and H.W. Lenstra. A sharpened version of the aanderaa-rosenberg conjecture. *Stichting Mathematisch Centrum. Zuivere Wiskunde. Stichting Mathematisch Centrum.*, 1974.
11. Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 512–527. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.
12. Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002.
13. Amit Chakrabarti and Subhash Khot. Improved lower bounds on the randomized complexity of graph properties. *Random Struct. Algorithms*, 30(3):427–440, 2007.
14. Andrew M Childs and Robin Kothari. Quantum query complexity of minor-closed graph properties. *SIAM Journal on Computing*, 41(6):1426–1450, 2012.
15. Rahul Chugh, Supartha Podder, and Swagato Sanyal. Decision tree complexity versus block sensitivity and degree. *arXiv preprint*, 2022. `arXiv:2209.08042`.
16. Yogesh Dahiya and Meena Mahajan. On (simple) decision tree rank. *Theoretical Computer Science*, page 114177, 2023. Preliminary version appeared in FSTTCS 2021.
17. Krishnamoorthy Dinesh and Jayalal Sarma. Alternation, sparsity and sensitivity: Bounds and exponential gaps. *Theoretical Computer Science*, 771:71–82, 2019.
18. Yevgeniy Dodis and Sanjeev Khanna. Space-time tradeoffs for graph properties. In *International Colloquium on Automata, Languages, and Programming*, pages 291–300. Springer, 1999.
19. Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

20 Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some boolean function complexity measures. *Combinatorica*, 36(3):265–311, 2016.

21 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for bpp. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–143. IEEE, 2017.

22 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM Journal on Computing*, 47(6):2435–2450, 2018.

23 Siyao Guo, Tal Malkin, Igor C Oliveira, and Alon Rosen. The power of negations in cryptography. In *Theory of Cryptography Conference*, pages 36–65. Springer, 2015.

24 Péter Hajnal. An Ω (n 4/3) lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(2):131–143, 1991.

25 Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. Variations on the sensitivity conjecture. *Theory of Computing*, pages 1–27, 2011.

26 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.

27 Valerie King. Lower bounds on the complexity of graph properties. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 468–476, 1988.

28 David Kirkpatrick. Determining graph properties from matrix representations. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 84–90, 1974.

29 Raghav Kulkarni and Avishay Tal. On fractional block sensitivity. *Chicago J. Theor. Comput. Sci*, 8:1–16, 2016.

30 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

31 Chengyu Lin and Shengyu Zhang. Sensitivity conjecture and log-rank conjecture for functions with small alternating numbers. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 51:1–51:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

32 Andrey A Markov. On the inversion complexity of a system of functions. *Journal of the ACM (JACM)*, 5(4):331–334, 1958.

33 Gatis Midrijanis. Exact quantum query complexity for total boolean functions. *arXiv*, 2004. `arXiv:quant-ph/0403168`.

34 Ashley Montanaro and Tobias Osborne. On the communication complexity of xor functions. *arXiv preprint*, 2009. `arXiv:0909.3392`.

35 Hiroki Morizumi. Limiting negations in formulas. In *International Colloquium on Automata, Languages, and Programming*, pages 701–712. Springer, 2009.

36 Hiroki Morizumi. Limiting negations in non-deterministic circuits. *Theoretical Computer Science*, 410(38-40):3988–3994, 2009.

37 Noam Nisan. Crew prams and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991.

38 Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational complexity*, 4(4):301–313, 1994.

39 Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995.

40 Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020.

41 Ran Raz and Pierre McKenzie. Separation of the monotone nc hierarchy. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 234–243. IEEE, 1997.

42 Ronald L Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3(3):371–384, 1976.

43 Arnold L Rosenberg. On the time required to recognize properties of graphs: A problem. *ACM SIGACT News*, 5(4):15–16, 1973.

44 Michael Saks and Avi Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 29–38. IEEE, 1986.

45 Miklos Santha and Christopher Wilson. Limiting negations in constant depth circuits. *SIAM Journal on Computing*, 22(2):294–302, 1993.

46 Amir Shpilka, Avishay Tal, and Ben Lee Volk. On the structure of boolean functions with small spectral norm. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 37–48, 2014.

47 Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: An extension of markov's theorem. *Information processing letters*, 90(1):15–20, 2004.

48 Avishay Tal. Properties and applications of boolean function composition. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 441–454, 2013.

49 Hing Yin Tsang. On boolean functions with low sensitivity. *manuscript*, 4:10–16, 2014.

50 Hing Yin Tsang, Chung Hoi Wong, Ning Xie, and Shengyu Zhang. Fourier sparsity, spectral norm, and the log-rank conjecture. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 658–667. IEEE, 2013.

51 G TURIN. The critical complexity of graph properties. *Inform. Process. Lm*, 18:151–153, 1984.

52 Joachim von Zur Gathen and James R Roche. Polynomials with two values. *Combinatorica*, 17(3):345–362, 1997.

53 Adam Wathieu. Exposition of the Kushilevitz function. *Technical report. Northwestern University, Computer Science Department*, NU-CS-2022-08, 2022.

54 Andrew Chi-Chih Yao. Lower bounds to randomized algorithms for graph properties. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 393–400. IEEE, 1987.

A Extended preliminaries

Throughout this section, we assume that $f : \{0,1\}^n \to \{0,1\}$is a generic Boolean function, and $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ a generic input to f. Restriction of f to a set $S \subseteq \{0,1\}^n$ is denoted by $f\mid_S$.

A.1 Complexity measures

▶ **Definition 23** (Sensitivity). *An index $i \in [n]$ is said to be a* sensitive *index (or a sensitive bit) for x is $f(x) \neq f(x^{\oplus i})$. The* sensitivity *of f on x, denoted by $\mathrm{s}(f,x)$, is defined as the number of sensitive bits for x. In other words,*

$$s(f,x) = |\{i \in [n] \mid f(x) \neq f(x^i)\}|.$$

The sensitivity *of f, denoted by $\mathrm{s}(f)$, is defined as*

$$\mathrm{s}(f) = \max_{x \in \{0,1\}^n} \mathrm{s}(f,x).$$

A set of indices $B \in [n]$ is called a *sensitive block* of x if $f(x) \neq f(x^{\oplus B})$.

▶ **Definition 24** (Block sensitivity). *The* block sensitivity *of f on input x, denoted by $\mathrm{bs}(f,x)$, is defined as the maximum number of disjoint sensitive blocks of x. For $b \in \{0,1\}$, the b-*block sensitivity *of f, denoted by $\mathrm{bs}^{(b)}(f)$, is defined as*

$$\mathrm{bs}^{(b)}(f) = \max_{x \in f^{-1}(b)} \mathrm{bs}(f,x).$$

The block sensitivity of f, *denoted by* $\mathrm{bs}(f)$, *is defined as*

$$\mathrm{bs}(f) := \max\{\mathrm{bs}^{(0)}(f), \mathrm{bs}^{(1)}(f)\}.$$

Let $\{B_1, \ldots, B_t\}$ be the set of all sensitive blocks of x. $\mathrm{bs}(f, x)$ is the value of the following integer linear program.

$$\begin{aligned} bs(f,x) = \quad & \max & & \sum_{j=1}^{t} w_j & \\ & \text{subject to} & & \sum_{j:B_j \ni i} w_j \leq 1 & \forall i \in [n] \\ & & & w_j \in \{0,1\} & \forall j \in [t] \end{aligned}$$

The value of the linear program obtained by relaxing the integrality constraint of the above program is called the fractional block sensitivity of f on input x, denoted by $\mathrm{fbs}(f, x)$.

▶ **Definition 25** (Fractional block sensitivity [29, 20]). *The* fractional block sensitivity *of f on x, denoted by* $\mathrm{fbs}(f, x)$, *is defined as*

$$\begin{aligned} fbs(f,x) = \quad & max & & \sum_{j=1}^{t} w_j & \\ & \textit{subject to} & & \sum_{j:B_j \ni i} w_j \leq 1 & \forall i \in [n] \\ & & & w_j \in [0,1] & \forall j \in [t] \end{aligned}$$

The fractional block sensitivity of f, *denoted by* $\mathrm{fbs}(f)$, *is defined as*

$$\mathrm{fbs}(f) = \max_{x \in \{0,1\}^n} \mathrm{fbs}(f, x).$$

▶ **Definition 26** (Certificate complexity, minimum certificate complexity and maximin certificate complexity). *A subcube C is called a* 0-certificate *(resp.* 1*-certificate) of f if $f|_C$ is the constant* 0 *(resp.* 1*) function. A subcube is called a* certificate *if it is a* 0*-certificate or a* 1*-certificate. The* certificate complexity of f on x, *denoted by* $\mathrm{C}(f, x)$, *is the smallest co-dimension of a certificate C that contains x. For a bit $b \in \{0, 1\}$ the* b-certificate complexity of f, *denoted by* $\mathrm{C}_b(f)$, *is defined as*

$$\mathrm{C}_b(f) = \max_{x \in \{0,1\}^n, f(x)=b} \mathrm{C}(f, x).$$

The certificate complexity of f, *denoted by* $\mathrm{C}(f)$, *is defined as,*

$$\mathrm{C}(f) = \max\{\mathrm{C}_0(f), \mathrm{C}_1(f)\}.$$

A minimum certificate *of f is defined to be a subcube of minimum co-dimension on which f is constant. The* minimum certificate complexity of f, *denoted by* $\mathrm{C}_{\min}(f)$, *is defined as the co-dimension of a minimum certificate. Equivalently,*

$$\mathrm{C}_{\min}(f) = \min_{x \in \{0,1\}^n} \mathrm{C}(f, x).$$

The maximin certificate complexity of f, *denoted by* $\mathrm{C}^*_{\min}(f)$, *is defined as,*

$$\mathrm{C}^*_{\min}(f) = \max_C \mathrm{C}_{\min}(f|_C).$$

Above, the maximum is over all subcubes of $\{0, 1\}^n$, and $f|_C$ is viewed as a Boolean function on the Boolean hypercube $\{0, 1\}^{[n]\setminus I}$.

We say that a set of indices $S \subseteq [n]$ forms a certificate for x, to mean that $\{y \in \{0,1\}^n \mid y_i = x_i \forall i \in S\}$ is a certificate.

▶ **Definition 27** (P_f and degree). *It is well known that every Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be represented by a unique multi-linear polynomial $P_f(x_1, \ldots, x_n)$ with real coefficients such that $P(x) = f(x)$ for all $x \in \{0,1\}^n$. The* exact degree of f*, or simply the degree of f, denoted by* $\deg(f)$*, is defined to be the degree of P_f.*

Identification of variables

Let $i, j \in [n]$ be distinct indices in $[n]$. Let $S := \{x \in \{0,1\}^n \mid x_i = x_j\}$. The restriction $f\mid_S$ of f to S is the function obtained from f by identifying variables x_i and x_j. $f\mid_S$ can be thought of as a Boolean function defined on the hypercube $\{0,1\}^{[n]\setminus\{j\}}$.

▶ **Definition 28.** *f is a* symmetric *Boolean function if $f(x)$ depends only on the Hamming weight of x, i.e., there exists a function $g : [n] \cup \{0\} \to \{0,1\}$ suh that $f(x) = g(|x|)$.*

The following proposition lists some well-known facts about various complexity measures [12, 29, 20].

▶ **Proposition 29.**
1. $s(f,x) \le \mathrm{bs}(f,x) \le \mathrm{fbs}(f,x) \le \mathrm{C}(f,x) \le \mathrm{D}(f)$.
2. $\mathrm{C}(f) = O(\mathrm{bs}(f)^2)$. *[37]*
3. $\mathrm{bs}(f) = O(\deg(f)^2)$. *[38]*
4. $\mathrm{D}(f) = O(\mathrm{C}(f) \cdot \mathrm{bs}(f))$. *[7]*
5. $\mathrm{D}(f) = O(\deg(f) \cdot \mathrm{bs}(f))$. *[33]*
6. $\deg(f) \le \mathrm{D}(f)$.
7. *Measures* $\mathrm{s}, \mathrm{bs}, \mathrm{fbs}, \mathrm{C}, \mathrm{C}^*_{\min}, \deg, \mathrm{D}$ *do not increase under restrictions to subcubes. That is, for any* $\mathsf{M} \in \{\mathrm{s}, \mathrm{bs}, \mathrm{fbs}, \mathrm{C}, \mathrm{C}^*_{\min}, \deg, \mathrm{D}\}$ *and any subcube C,* $\mathsf{M}(f\mid_C) \le \mathsf{M}(f)$.[8]
8. $\mathrm{bs}, \deg, \mathrm{C}$ *and* D *do not increase under identification of variables. That is, for* $\mathsf{M} \in \{\mathrm{bs}, \deg \mathrm{C}, \mathrm{D}\}$*, distinct $i, j \in [n]$ and $S = \{x \in \{0,1\}^n \mid x_i = x_j\}$,* $\mathsf{M}(f\mid_S) \le \mathsf{M}(f)$.[9]
9. *Let f be non-constant and symmetric. Then* $\deg(f) = n - o(n)$ *[52] and* $\mathrm{bs}(f) = \mathrm{s}(f) \ge \lceil \frac{n+1}{2} \rceil$ *[51].*

A.2 Communication complexity

In the two-party communication model (introduced by Yao [4]), two parties Alice and Bob jointly hold an input to a Boolean function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$. There is a bi-partition of the set of input indices $[n]$ into two parts of sizes n_1 and n_2, say, where $n_1 + n_2 = n$. Alice and Bob respectively hold the input bits that correspond to the indices in these two parts. Let $x^{(1)}$ and $x^{(2)}$ be the input strings held by Alice and Bob respectively. Alice and Bob are interested in computing $f(x^{(1)}, x^{(2)})$, and they are allowed to exchange bits in an interactive fashion via a communication channel. The objective is to jointly compute $f(x^{(1)}, x^{(2)})$ by exchanging as few bits as possible. A set of rules that the parties follow to determine the messages that they send to each other in each step is called a protocol. A protocol for f is correct if for every input $(x^{(1)}, x^{(2)})$, at the end of the interaction, a party outputs $f(x^{(1)}, x^{(2)})$.

[8] Note that the same cannot be said about $\mathrm{C}_{\min}$.
[9] The same cannot be said about s.

The cost of a protocol Π on an input $(x^{(1)}, x^{(2)}) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}$, denoted by $\mathsf{cost}(\Pi, (x^{(1)}, x^{(2)}))$, is the number of bits exchanged by Π on input $(x^{(1)}, x^{(2)})$. The deterministic communication complexity of f, denoted by $\mathsf{CC}(f)$, is defined as follows:

$$\mathsf{CC}(F) = \min_{\Pi} \max_{(x^{(1)}, x^{(2)}) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}} \mathsf{cost}(\Pi, (x^{(1)}, x^{(2)})).$$

where the minimum is over all correct protocols of f. The bi-partition of the inputs in the definition of CC is implicit and will be clear from the context. See Section 2 and textbooks by Rao and Yehudayoff [40] and by Kushilevitz and Nisan [30] for a comprehensive introduction to the subject.

B Functions with bounded alternation number

For the sake of completeness, we reproduce the proofs of Lemma 7 and Lemma 8 here. Our proofs deviate slightly from the original ones.

Proof of Lemma 7. We will show that Algorithm 1 makes at most $\mathrm{C}^*_{\min}(f)\deg(f)$ queries.

First we show that every certificate of f must intersect every maximum degree monomial of the unique multilinear polynomial P_f that represents f. Suppose not, and there exists a certificate $C = \{y \in \mathcal{D} \mid y_i = a_i \forall i \in S\}$ and a maximum monomial M of P_f, such that $S \cap M = \emptyset$. For each $i \in S$, substitute a_i for y_i. since M is a leading monomial and disjoint from S, the resultant polynomial is non-constant, that contradicts the assumption that C is a certificate.

It follows that after every iteration of the *while* loop, $\deg(g)$ drops by at least 1. Hence the number of iterations of the loop is at most $\deg(f)$. Since we make at most $\mathrm{C}^*_{\min}(f)$ queries in each iteration, the theorem follows. ◀

Proof of Lemma 8 . Let $\mathrm{alt}(f) \le k$. We first show that $\mathrm{C}_{\min}(f) \le k \cdot \mathrm{bs}(f)$ and $\mathrm{C}_{\min}(f) \le k \cdot \deg(f)$. Note that, alternation number and block sensitivity does not increase udser restrictions (Observation 14 and Proposition 29 (7)), therefore we will have that $\mathrm{C}^*_{\min}(f) \le k \cdot \mathrm{bs}(f)$ and $\mathrm{C}^*_{\min}(f) \le k \cdot \deg(f)$.

(Part (i): $\mathrm{C}_{\min}(f) \le k \cdot \mathrm{bs}(f)$) Consider the subcube returned by Algorithm 2.

Algorithm 2 Algorithm 2 of Lemma 8.

1 **Initialize** $S^{(0)} \leftarrow \emptyset, C^{(0)} \leftarrow \{0,1\}^n, t \leftarrow 1$, $x^{(0)} \leftarrow 0^n$.
2 **while** $f\,|_{C^{(t-1)}}$ *is not a constant function* **do**
3 Let $x^{(t)} > x^{(t-1)}$ be a point with minimum Hamming weight such that $f(x^{(t)}) \neq f(x^{(t-1)})$.
4 $S^{(t)} \leftarrow \{i \in [n] \mid x_i^{(t)} = 1\}$.
5 $C^{(t)} \leftarrow \{z \in \{0,1\}^n \mid z_i = 1 \text{ for all } i \in S^{(t)}\}$.
6 $t \leftarrow t + 1$.
7 **end**
8 Return $C^{(t-1)}$.

We first show that the step 3 of Algorithm 2 is well-defined: there exists an $x^{(t)}$ for the algorithm to choose. Since f is non-constant on $C^{(t-1)}$ and $x^{(t-1)} \in C^{(t-1)}$, therefore there exists another point $x^{(t)} \in C^{(t-1)}$ such that $f(x^{(t)}) \neq f(x^{(t-1)})$. Next, note that $x_i^{(t-1)} = 1$ for all $i \in S^{(t-1)}$ and $x_i^{(t-1)} = 0$ otherwise. Thus, for any other point $x^{(t)} \in C^{(t-1)}$, $x^{(t)} > x^{(t-1)}$. Thus, an $x^{(t)}$ is guaranteed to exist for the algorithm to pick.

Let ℓ be the number of iterations of Algorithm 2. Thus, $C^{(\ell)}$ is the subcube returned by Algorithm 2. Clearly, $f|_{C^{(\ell)}}$ is a constant function; thus Algorithm 2 indeed returns a certificate with co-dimension $\mathsf{codim}(C^{(\ell)}) = |S^{(\ell)}|$. Now by the choice of $x^{(t)}$, any index j such that $x_j^{(t)} = 1$ and $x_j^{(t-1)} = 0$ is sensitive for $x^{(t)}$. Thus the number of such locations is at most $s(f)$. Thus for each $t = 1, \ldots, \ell$, $|S^{(t)}| \leq |S^{(t-1)}| + \mathrm{s}(f)$, giving us $\mathsf{codim}(C^{(\ell)}) = |S^{(\ell)}| \leq \ell \cdot \mathrm{s}(f) \leq \ell \cdot \mathrm{bs}(f)$ (Proposition 29 (1)).

Now, it follows from Observation 13 that there exists a monotone path from 0^n to $x^{(\ell)}$ that passes through each $x^{(t)}$ for $t = 1, \ldots, \ell - 1$. The alternation number of the path is at least ℓ, giving us $\ell \leq k$. Hence $\mathsf{codim}(C^{(\ell)}) \leq k \cdot \mathrm{bs}(f)$.

(Part (ii): $\mathrm{C}_{\min}(f) \leq k \cdot \deg(f)$) Consider the subcube returned by Algorithm 3.

Algorithm 3 Algorithm 3 of Lemma 8.

1 **Initialize** $S^{(0)} \leftarrow \emptyset, C^{(0)} \leftarrow \{0,1\}^n, t \leftarrow 1, x^{(0)} \leftarrow 0^n$.
2 **while** $f|_{C^{(t-1)}}$ *is not a constant function* **do**
3 Let M be a maximal monomial of $P_{f|_{C^{(t-1)}}}$.
4 Let $x^{(t)} > x^{(t-1)}$ be a point such that $x_i^{(t)} = 0$ for all $i \notin M \cup S^{(t-1)}$ and $f(x^{(t)}) \neq f(x^{(t-1)})$.
5 $S^{(t)} \leftarrow \{i \in [n] \mid x_i^{(t)} = 1\}$.
6 $C^{(t)} \leftarrow \{z \in \{0,1\}^n \mid z_i = 1 \text{ for all } i \in S^{(t)}\}$.
7 $t \leftarrow t + 1$.
8 **end**
9 Return $C^{(t-1)}$.

First, we show that step 4 of Algorithm 3 is well defined: there indeed exists an $x^{(t)}$ for the algorithm to choose. $P_{f|_{C^{(t-1)}}}$ is a polynomial on variables $\{x_i \mid i \notin S^{(t-1)}\}$ obtained by substituting 1 for the variables x_i for all $i \in S^{(t-1)}$. Now, substitute 0 for all variables outside of M in $P_{f|_{C^{(t-1)}}}$. Since M is a maximal monomial of $P_{f|_{C^{(t-1)}}}$, we are left with a non-zero polynomial P', say, on the remaining set of variables $\{x_i \mid i \in M \setminus S^{(t-1)}\}$. Since P' is non-constant, therefore there exists a non-zero input $z = (z_i)_{i \in M \setminus S^{(t-1)}}$ such that

$$P'(z) \neq P'(0^{M \setminus S^{(t-1)}}) = P_{f|_{C^{(t-1)}}}(0^{[n] \setminus S^{(t-1)}}) = P_f(x^{(t-1)}) = f(x^{(t-1)}). \tag{1}$$

Now, define $x^{(t)}$ as follows:

$$x_i^{(t)} = \begin{cases} z_i & \text{if } i \in M \setminus S^{(t-1)}, \\ 1 & \text{if } i \in S^{(t-1)}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus $P'(z) = P_f(x^{(t)}) = f(x^{(t)})$. Furthermore, since $x_i^{(t-1)} = 1$ if $i \in S^{(t-1)}$ and $x_i^{(t-1)} = 0$ otherwise, we have that $x^{(t)} > x^{(t-1)}$. Equation (1) now lets us conclude that $x^{(t)}$ satisfies the criteria to be chosen by the algorithm in step 4.

Now we analyze the subcube output by Algorithm 3. Let ℓ be the number of iterations of Algorithm 3. Hence $C^{(\ell)}$ is the subcube returned by Algorithm 3. Clearly, $f|_{C^{(\ell)}}$ is a constant function; thus Algorithm 3 indeed returns a certificate with co-dimension $\mathsf{codim}(C^{(\ell)}) = |S^{(\ell)}|$. Now by the choice of $x^{(t)}$, $x^{(t)}$ has at most $|M| \leq \deg(f)$ 1s more than $x^{(t-1)}$. Thus for each $t = 1, \ldots, \ell$, $|S^{(t)}| \leq |S^{(t-1)}| + \deg(f)$, giving us $\mathsf{codim}(C^{(\ell)}) = |S^{(\ell)}| \leq \ell \cdot \deg(f)$. Similar to Part (i), it follows that $\ell \leq k$, concluding the proof. ◀

FPT Approximations for Packing and Covering Problems Parameterized by Elimination Distance and Even Less

Tanmay Inamdar
University of Bergen, Norway

Lawqueen Kanesh
Indian Institute of Technology Jodhpur, India

Madhumita Kundu
University of Bergen, Norway

M. S. Ramanujan
University of Warwick, Coventry, UK

Saket Saurabh
Institute of Mathematical Sciences, Chennai, India
University of Bergen, Norway

Abstract

For numerous graph problems in the realm of parameterized algorithms, using the size of a smallest deletion set (called a modulator) into well-understood graph families as parameterization has led to a long and successful line of research. Recently, however, there has been an extensive study of structural parameters that are potentially much smaller than the modulator size. In particular, recent papers [Jansen et al. STOC 2021; Agrawal et al. SODA 2022] have studied parameterization by the size of the modulator to a graph family $\mathcal{H}$ ($\mathbf{mod}_{\mathcal{H}}(\cdot)$), elimination distance to $\mathcal{H}$ ($\mathbf{ed}_{\mathcal{H}}(\cdot)$), and $\mathcal{H}$-treewidth ($\mathbf{tw}_{\mathcal{H}}(\cdot)$). These parameters are related by the fact that $\mathbf{tw}_{\mathcal{H}}$ lower bounds $\mathbf{ed}_{\mathcal{H}}$, which in turn lower bounds $\mathbf{mod}_{\mathcal{H}}$. While these new parameters have been successfully exploited to design fast exact algorithms their utility (especially that of $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$) in the context of approximation algorithms is mostly unexplored.

The conceptual contribution of this paper is to present novel algorithmic meta-theorems that expand the impact of these structural parameters to the area of FPT Approximation, mirroring their utility in the design of exact FPT algorithms. Precisely, we show that if a covering or packing problem is definable in Monadic Second Order Logic and has a property called Finite Integer Index (FII), then the existence of an FPT Approximation Scheme (FPT-AS, i.e., $(1 \pm \epsilon)$-approximation) parameterized by $\mathbf{mod}_{\mathcal{H}}(\cdot), \mathbf{ed}_{\mathcal{H}}(\cdot)$, and $\mathbf{tw}_{\mathcal{H}}(\cdot)$ is in fact equivalent. As a consequence, we obtain FPT-ASes for a wide range of covering, packing, and domination problems on graphs with respect to these parameters. In the process, we show that several graph problems, that are W[1]-hard parameterized by $\mathbf{mod}_{\mathcal{H}}$, admit FPT-ASes not only when parameterized by $\mathbf{mod}_{\mathcal{H}}$, but even when parameterized by the potentially much smaller parameter $\mathbf{tw}_{\mathcal{H}}(\cdot)$. In the spirit of [Agrawal et al. SODA 2022], our algorithmic results highlight a broader connection between these parameters in the world of approximation. As concrete exemplifications of our meta-theorems, we obtain FPT-ASes for well-studied graph problems such as Vertex Cover, Feedback Vertex Set, Cycle Packing and Dominating Set, parameterized by $\mathbf{tw}_{\mathcal{H}}(\cdot)$ (and hence, also by $\mathbf{mod}_{\mathcal{H}}(\cdot)$ or $\mathbf{ed}_{\mathcal{H}}(\cdot)$), where $\mathcal{H}$ is any family of minor free graphs.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases FPT-AS, $\mathcal{F}$-Deletion, Packing, Elimination Distance, Elimination Treewidth

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.28

Related Version *Full Version*: https://arxiv.org/abs/2310.03469

© Tanmay Inamdar, Lawqueen Kanesh, Madhumita Kundu, M. S. Ramanujan, and Saket Saurabh; licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 28; pp. 28:1–28:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Tanmay Inamdar*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416).
M. S. Ramanujan: Supported by Engineering and Physical Sciences Research Council (EPSRC) grants EP/V007793/1 and EP/V044621/1.
Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416), and Swarnajayanti Fellowship (No. DST/SJF/MSA01/2017-18).

1 Introduction

One of the most widely studied graph problems in the area of parameterized complexity is the $\mathcal{F}$-Vertex Deletion problem, where the input is a graph G and a number k and the goal is – "Compute a set of at most k vertices whose deletion places the resulting graph in the graph family $\mathcal{F}$ or correctly conclude that such a set does not exist." A solution to an instance of $\mathcal{F}$-Vertex Deletion is called a *modulator* into $\mathcal{F}$ and there are numerous results in parameterized complexity on exploiting modulators into various graph families to design algorithms. Much of this research has been motivated by the fact that inputs that have modulators of small size into $\mathcal{F}$ turn out to be tractable for many problems that are NP-complete in general while being polynomial-time solvable on $\mathcal{F}$. In other words, it is possible to take efficient algorithms for some problems on graphs in $\mathcal{F}$, and lift them to efficient algorithms for these problems on graphs that are not necessarily in $\mathcal{F}$, but have a small vertex modulator into $\mathcal{F}$, i.e., graphs that are "close" to $\mathcal{F}$. This leads to fixed-parameter algorithms for these problems (i.e., running time bounded by $f(k) \cdot n^{O(1)}$, where k is the modulator size and n is the input size) under these parameterizations. Using the size of the smallest vertex modulator of a graph into tractable graph families or "the distance from triviality" methodology [12] has therefore become a rich source of interesting and useful parameters for graph problems over the last two decades.

In light of the success of this line of research, recent years have seen a shift towards identifying and exploring the power of "hybrid" parameters that are upper bounded by the modulator size as well as certain graph-width measures and can be arbitrarily (and simultaneously) smaller than both the modulator size and these graph-width measures. Two specific parameters studied in this line of research are: $\mathcal{H}$-elimination distance and $\mathcal{H}$-treewidth of G. The $\mathcal{H}$-elimination distance of a graph G (denoted $\mathbf{ed}_{\mathcal{H}}(G)$) was introduced by Bulian and Dawar [2] and roughly speaking, it expresses the number of rounds needed to obtain a graph in $\mathcal{H}$ by removing one vertex from every connected component in each round. We refer the reader to Section 2 for a more formal definition. Note that $\mathbf{ed}_{\mathcal{H}}(G)$ (respectively, $\mathbf{tw}_{\mathcal{H}}(G)$) can be arbitrarily smaller than both $\mathbf{mod}_{\mathcal{H}}(G)$ and the treedepth of G (respectively, the treewidth of G). For example, let $\mathcal{H}$ be the (infinite) family of complete graphs, and let G be a graph that contains a vertex v that is adjacent to some (non-empty) subset of vertices from a clique of size t, for some $t \geq 1$. Then, $\mathbf{tw}(G) = t - 1$, whereas $\mathbf{tw}_{\mathcal{H}}(G) = 0$ (in fact, even $\mathbf{ed}_{\mathcal{H}}(G) = 0$). Further, $\mathbf{tw}_{\mathcal{H}}(G)$ itself can be arbitrarily smaller than $\mathbf{ed}_{\mathcal{H}}(G)$ (see [9, 15] for some examples).

Recent work by Agrawal et al. [1] and Jansen et al. [15] show that for many basic graph problems in the literature and well-understood graph families $\mathcal{F}$, one can indeed obtain FPT algorithms parameterized by $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$, thus expanding the notion of useful distance from triviality to encompass these parameters as well. Agrawal et al. [1] also showed a tight connection between $\mathbf{mod}_{\mathcal{H}}, \mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$ by showing that for these problems, having an FPT algorithm parameterized by $\mathbf{mod}_{\mathcal{H}}$ was sufficient to obtain FPT algorithms parameterized

by the other two "smaller" parameters. In addition to these results, there is also some recent work on computing $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$, where $\mathcal{H}$ is the family of bipartite graphs [14], or a minor-closed family [19].

Despite these leaps in our understanding of the parameterized complexity of many problems, limitations remain. For instance, by requiring that the problem be FPT parameterized by $\mathbf{mod}_{\mathcal{H}}$, we are implicitly requiring that the problem be polynomial-time solvable on the class $\mathcal{H}$. This rules out meaningful results for many basic problems and established graph families $\mathcal{H}$. For instance, it is not interesting to study Vertex Cover parameterized by $\mathbf{mod}_{\mathcal{H}}$ when $\mathcal{H}$ is the class of planar graphs since Vertex Cover is NP-complete on planar graphs [10]. However, it is efficiently approximable on planar graphs (i.e., even has an Efficient PTAS) [7]. This state of the art brings us to the following two natural questions and is the main motivation behind this work.

Question 1: Can good *approximation algorithms* for a problem on the class $\mathcal{H}$ be used to obtain good FPT approximation algorithms for the same problem parameterized by $\mathbf{mod}_{\mathcal{H}}$?

Question 2: Could one obtain positive answers to the above question, but for the parameters $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$?

The notion of an FPT approximation algorithm is easily motivated by the simultaneous existence of fixed-parameter intractability results as well as polynomial-time inapproximability results for numerous problems in the literature. Hence, the topic of FPT approximation, has been an extremely active area of research in the last decade. For a comprehensive survey of the state of the art, we refer the reader to the survey by Feldman et al. [5].

We note that we are not the first to study Question 1. In Marx's classic survey [18] on parameterized approximation, he outlines an FPT approximation algorithm for Chromatic Number parameterized by the modulator to planar graphs. More recently, Demaine et al. [4] studied this question systematically, albeit restricted to polynomial-time approximation. They developed a general theorem that gives sufficient conditions on the problem in order to guarantee an affirmative answer to this question when the class $\mathcal{H}$ has bounded treewidth or arboricity, but in their context, the approximation ratio of the algorithms depends on $\mathbf{mod}_{\mathcal{H}}$. However, Question 1 in the setting of FPT approximation is largely unexplored. To the best of our knowledge, Question 2 has not been considered in the literature and this is the main focus of our paper.

1.1 Our contributions

The main conceptual message of the paper is the following meta-result (stated informally):

If a problem can be captured by a very expressible logic fragment (i.e, Counting Monadic Second Order Logic (CMSO)) and has an appropriate graph replacement subroutine (i.e., has Finite Integer Index (FII)) and fulfills a few other mild requirements, then the existence of an FPT-AS for the problem parameterized by any of the three parameters $\mathbf{mod}_{\mathcal{H}}, \mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$ is equivalent to each other.

The formal version of our first meta-theorem in the context of the well-studied family of vertex-deletion problems is given below. Say that a family of graphs is *well-behaved* if it is hereditary and closed under disjoint union.

▶ **Theorem 1.** *Let $\mathcal{H}, \mathcal{F}$ be well-behaved families of graphs, where $\mathcal{F}$ is CMSO-definable. Suppose $\Pi =$ Vertex Deletion to $\mathcal{F}$ has* FII. *Then, the following statements are equivalent.*

1. Π *admits an* FPT-AS *parameterized by* $\mathbf{mod}_{\mathcal{H}}(\cdot)$ *and* ϵ.
2. Π *admits an* FPT-AS *parameterized by* $\mathbf{ed}_{\mathcal{H}}(\cdot)$ *and* ϵ.
3. Π *admits an* FPT-AS *parameterized by* $\mathbf{tw}_{\mathcal{H}}(\cdot)$ *and* ϵ.

In the full version of the paper, we also prove meta-theorems similar to Theorem 1 where Π is $\mathcal{F}$-Subgraph Packing (i.e., pack maximum number of vertex-disjoint subgraphs isomorphic to graphs in $\mathcal{F}$) or $\mathcal{F}$-Minor Packing (i.e., pack maximum number of vertex-disjoint minor-models of graphs in $\mathcal{F}$).

In order to invoke these equivalence theorems, we first show that a wide range of graph problems, that are W[1]-hard (or para-NP hard) parameterized by $\mathbf{mod}_{\mathcal{H}}$, admit FPT Approximation Schemes (FPT-ASes, i.e., $(1 \pm \epsilon)$-approximation) when parameterized by $\mathbf{mod}_{\mathcal{H}}$. Hence, as corollaries of our meta-results, we also get the first FPT-ASes for the following (non-exhaustive list of) problems parameterized by $\mathbf{tw}_{\mathcal{H}}$ (and hence, also by $\mathbf{ed}_{\mathcal{H}}$). When $\mathcal{H}$ is apex-minor free, we obtain this result for Independent Set, Triangle Packing (note that the latter is a special case of $\mathcal{F}$-Subgraph Packing). When $\mathcal{H}$ is minor free, we are able to infer the same result for Vertex Cover, Feedback Vertex Set, Cycle Packing (and more generally, $\mathcal{F}$-Minor Packing). 5

Finally, we identify Dominating Set, Connected Dominating Set, and Connected Vertex Cover as interesting special cases that are not covered by the framework developed above and give purpose-built FPT-ASes for them parameterized by $\mathbf{mod}_{\mathcal{H}}$ and then also prove an equivalence theorem in the same spirit as Theorem 1. Among these three problems, we give the details for Dominating Set, and defer the results for the connectivity constrained problems to the full version of the paper.

In summary, our work highlights many natural problems for which the answer to Question 2 is affirmative. Finally, we note that in Theorem 1 and its variants, due to their generality, we only obtain *non-uniform* FPT-ASes. In Conclusion, we briefly discuss certain scenarios where it may be possible to obtain uniform FPT-ASes.

1.2 Our techniques

We first summarize the technique behind our FPT-ASes parameterized by $\mathbf{mod}_{\mathcal{H}}$, which we dub "Bucket vs Ocean". Recall that this is required in order to invoke Theorem 1 to get our eventual results, i.e., FPT-ASes parameterized by $\mathbf{tw}_{\mathcal{H}}$ and $\mathbf{ed}_{\mathcal{H}}$. We remark that throughout the paper, we assume that our FPT-AS parameterized by, say $\mathbf{mod}_{\mathcal{H}}$ (resp. $\mathbf{ed}_{\mathcal{H}}, \mathbf{tw}_{\mathcal{H}}$) is also provided with a modulator to $\mathcal{H}$ (resp. $\mathcal{H}$-elimination/tree decomposition) of the appropriate size (resp. width). For $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$, one can use algorithms from [9, 15, 1, 14] to compute the appropriate decompositions exactly. Alternatively, one can use the recent result of [16] to compute constant approximations thereof.

Consider a special case of Vertex Deletion to $\mathcal{F}$, say Vertex Cover. Consider an H-minor free graph family $\mathcal{H}$, where VC admits an EPTAS. Consider a graph G and $M \subseteq V(G)$ of size $\mathbf{mod}_{\mathcal{H}}(G)$ such that $G - M \in \mathcal{H}$. We use the known EPTAS on $G - M$ to compute an $(1 + \epsilon/2)$-approximate solution to VC, call it S. Next, we compare $|S|$ with $|M|$, and consider two cases. If $|M| < \epsilon/3 \cdot |S| \leq (1 + \epsilon/2) \cdot \mathsf{OPT}(G - M)$, i.e., $\mathsf{OPT}(G - M)$ is

like an "ocean", compared to a "bucket" of water that is M, then we can add the bucket to the ocean "for free", i.e., $|S| + |M| \leq (1+\epsilon) \cdot \mathsf{OPT}(G - M) \leq (1+\epsilon) \cdot \mathsf{OPT}(G)$. Otherwise, if $|M|$ and $|S|$ are comparable, then $\mathsf{OPT}(G) \leq 3\mathbf{mod}_{\mathcal{H}}(G)/\epsilon$, in which case we can use the FPT algorithm for Vertex Cover, which is in fact in FPT in $\mathbf{mod}_{\mathcal{H}}$ and ϵ. This is the high level idea behind our FPT-ASes parameterized by $\mathbf{mod}_{\mathcal{H}}$, although one has to overcome several problem-specific challenges to make it work for the other problems.

A more sophisticated version of the idea also turns out to be useful in proving the equivalence theorem. Again, let us consider the example of Vertex Cover and $\mathcal{H}$ being an H-minor free graph family. Having armed ourselves with an FPT-AS parameterized by $\mathbf{mod}_{\mathcal{H}}$ as in the previous paragraph, now our task is to generalize to $\mathbf{tw}_{\mathcal{H}}$. To this end, we consider each bag $\chi(t)$ corresponding to a node t in the $\mathcal{H}$-tree decomposition (defined formally in the next section), which consists of $\ell \leq \mathbf{tw}_{\mathcal{H}}(G) + 1$ vertices, say R_t, which locally act like a modulator to a disjoint "base graph", say $G[H_t] \in \mathcal{H}$. We use the same "Bucket vs Ocean" idea to classify each node as *good* or *bad*. A node t is *good*, if a $(1+\epsilon)$-approximate solution S_t for $G[H_t]$ is like an "ocean" compared to R_t. In this case, we can again add R_t to S_t "for free". Otherwise, we say that a node t is bad, if $\mathsf{OPT}(G[H_t])$ is bounded by $3\ell/\epsilon$. From here, our task is to reduce the treewidth of the graph induced by the vertices in the bags of bad nodes. To this end, we use the FII property as well as the FPT-AS to perform a series of "graph replacements" in each bad node, where in each iteration, we replace the corresponding bag with an "equivalent subgraph" of size bounded by a function of ℓ and ϵ. At the end of this procedure, the resulting graph has treewidth bounded by a function of ℓ and ϵ, and we can use the CMSO-definability and Courcelle's theorem [3] to find an optimal solution (that can then be translated back to the original instance). It can be shown that the resulting solution is a $(1+\epsilon)$-approximate solution for the entire graph, and the whole algorithm runs in time FPT in $\mathbf{tw}_{\mathcal{H}}$ and ϵ.

Due to space constraints some of the results and figures are moved to appendix and can be found in the full version of the paper [13].

2 Preliminaries

For an instance $\mathcal{I}$ of an optimization problem Π, we denote the value/size of an optimal solution by $\mathsf{OPT}_{\Pi}(\mathcal{I})$, and may omit the subscript if the problem is clear from the context. An algorithm that, for any feasible instance of minimization (resp. maximization) problem Π, returns a solution of cost/size at most (resp. at least) $\alpha \cdot \mathsf{OPT}$, is called an α-approximation. We say that a minimization (resp. maximization) problem Π admits an FPT-AS (FPT-Approximation Scheme) parameterized by a parameter t and ϵ, if there exists an $(1+\epsilon)$-approximation (resp. $(1-\epsilon)$-approximation) algorithm with running time of the form $f(t,\epsilon) \cdot |\mathcal{I}|^{\mathcal{O}(1)}$, for some computable function f. Similarly, we say that Π admits an EPTAS (efficient polynomial-time approximation scheme), if there exists a $(1+\epsilon)$-approximation (resp. a $(1-\epsilon)$-approximation) algorithm with running time of the form $g(\epsilon) \cdot |\mathcal{I}|^{\mathcal{O}(1)}$, for some computable function g.

Let $\mathcal{G}$ denote the family of all graphs. We say that a family $\mathcal{F} \subseteq \mathcal{G}$ of graphs is *hereditary* if for any graph $G \in \mathcal{F}$, every induced subgraph G' of G also belongs to $\mathcal{F}$. We say that a family $\mathcal{F} \subseteq \mathcal{G}$ is *closed under disjoint union* if for any $G_1, G_2 \in \mathcal{F}$, the graph G obtained by taking the disjoint union of G_1 and G_2, also belongs to $\mathcal{F}$.

Graph Decompositions

The following definitions are borrowed from [1]. For a graph G, $\mathbf{mod}_{\mathcal{H}}(G)$ denotes the size of a smallest vertex set S such that $G - S \in \mathcal{H}$. If $G - S \in \mathcal{H}$, then S is called a *modulator* to $\mathcal{H}$.

▶ **Definition 2.** *For a graph family $\mathcal{H}$, an $\mathcal{H}$-*elimination decomposition *of G is a triple (T, χ, L), where T is a rooted forest, $\chi : V(T) \to 2^{V(G)}$, and $L \subseteq V(G)$ such that:*

1. *For each internal node $t \in V(T)$, we have that $|\chi(t)| \leq 1$, and $\chi(t) \subseteq V(G) \setminus L$.*
2. *The sets $(\chi(t))_{t \in V(T)}$ form a partition of $V(G)$.*
3. *For each leaf t in T, we have $\chi(t) \subseteq L$, such that the graph $G[\chi(t)]$, called a base component, belongs to $\mathcal{H}$. Furthermore, $(\chi(t))_{leaf\ t}$ forms a partition of L.*
4. *For each edge $uv \in V(G)$, if $u \in \chi(t_1)$, and $v \in \chi(t_2)$, then t_1 and t_2 are in ancestor-descendant relation in T.*

The depth of a rooted tree T is the maximum number of edges on a root-to-leaf path in T. We refer to the union of base components as the set of base vertices. The $\mathcal{H}$-elimination distance of G, denoted as $\mathbf{ed}_{\mathcal{H}}(G)$, is the minimum depth of an $\mathcal{H}$-elimination forest for G. Note that a pair (T, χ) is a (standard) elimination forest of $\mathcal{H}$ is a class of empty graphs, i.e., the base components are empty. In this case, $\mathbf{ed}_{\mathcal{H}}(G)$ is known as the *treedepth* of G, and is denoted as $\mathbf{td}(G)$.

Just like the notion of $\mathcal{H}$-elimination decomposition generalizes the notion of elimination forest, the following is an analogous generalization of the notion of tree decomposition.

▶ **Definition 3.** *For a graph family $\mathcal{H}$, an $\mathcal{H}$-*tree decomposition *of a graph G is a triple (T, χ, L), where T is a rooted tree, $\chi : V(T) \to 2^{V(G)}$, and $L \subseteq V(G)$, such that:*

1. *For each $v \in V(G)$, the nodes $\{t \in V(T) : v \in \chi(t)\}$ induce a non-empty connected subtree in T.*
2. *For each edge $uv \in E(G)$, there is a node $t \in V(G)$ with $\{u, v\} \in \chi(t)$.*
3. *For each vertex $v \in L$, there is a* unique leaf *$t \in V(T)$ for which $v \in \chi(t)$.*
4. *For each leaf node $t \in V(T)$, the graph $G[\chi(t) \cap L] \in \mathcal{H}$.*

The *width* of an $\mathcal{H}$-tree decomposition is defined as $\max\left\{0, \max_{t \in V(T)} |\chi(t) \setminus L| - 1\right\}$. The $\mathcal{H}$-treewidth of G, denoted by $\mathbf{tw}_{\mathcal{H}}(G)$, is the minimum width of an $\mathcal{H}$-tree decomposition of G. The connected components of $G[L]$ are called base components, and the vertices in L are called base vertices.

A pair (T, χ) is a (standard) *tree decomposition* if $(T, \chi, \emptyset)$ satisfies all conditions of an $\mathcal{H}$-decomposition, where the choice of $\mathcal{H}$ is irrelevant.

Additional preliminaries can be found in the full version of the paper.

3 Vertex Deletion to $\mathcal{F}$

In this section we give our "first equivalence" result. Towards that, first in Section 3.2, we design concrete FPT-ASes for several Vertex Deletion to $\mathcal{F}$ type problems (e.g., Vertex Cover, Feedback Vertex Set parameterized by $\mathbf{mod}_{\mathcal{H}}$, where $\mathcal{H}$ is an apex-minor family of graphs. Then, in Section 3.3 we prove our main equivalence theorem, which lets us extend the previous FPT-ASes parameterized by $\mathbf{mod}_{\mathcal{H}}(\cdot)$ to $\mathbf{ed}_{\mathcal{H}}(\cdot)$, and $\mathbf{tw}_{\mathcal{H}}(\cdot)$.

3.1 Preliminaries for Vertex Deletion to $\mathcal{F}$

We focus on the following problem, specifically on the case where $\mathcal{F}$ is some fixed well-behaved family of graphs.

In Vertex Deletion to $\mathcal{F}$ problem, we are given an instance (G, k), where $G = (V, E)$ is a graph, and $k \geq 0$ is an integer. And the question is does there exist a subset $S \subseteq V(G)$ of size at most k, such that $G - S \in \mathcal{F}$?

Optimization variant

Let $\mathcal{F}$ be a well-behaved family of graphs. For a graph G, a set $S \subseteq V(G)$ is said to be a *solution* to Vertex Deletion to $\mathcal{F}$, if $G - S \in \mathcal{F}$. In the optimization variant of Vertex Deletion to $\mathcal{F}$, we want to find a solution of the smallest cardinality. By slightly abusing the notation, we will use Vertex Deletion to $\mathcal{F}$ to refer to the decision as well as the optimization version, and will only disambiguate when strictly necessary. Note that assuming $\mathcal{F}$ contains at least one graph, the hereditary property implies that the empty graph belongs to $\mathcal{F}$, which means that $V(G)$ is always a solution to Vertex Deletion to $\mathcal{F}$. The proof of the following observation is straightforward and can be found in the full version.

▶ **Observation 4.** *Let $\mathcal{F}$ be a well-behaved family, and let $\Pi =$ Vertex Deletion to $\mathcal{F}$. Then, for any graph G and any subset $S \subseteq V(G)$, it holds that*

$$\mathsf{OPT}_\Pi(G - S) \leq \mathsf{OPT}_\Pi(G) \leq \mathsf{OPT}_\Pi(G - S) + |S|.$$

We also note that Vertex Deletion to $\mathcal{F}$ is *self-reducible*, i.e., an algorithm for the decision version can be used to actually find a solution set. We define this notion formally and give the proof in the full version.

3.2 FPT-ASes for Vertex Deletion to $\mathcal{F}$ Parameterized by $\mathrm{mod}_{\mathcal{H}}(\cdot)$

Let $\mathcal{H}, \mathcal{F}$ be well-behaved families of graphs, such that $\Pi =$ Vertex Deletion to $\mathcal{F}$ admits an EPTAS for (the optimization variant of) Π on any $G \in \mathcal{H}$. Then, we show how to design an FPT-AS for Π parameterized by $\mathbf{mod}_{\mathcal{H}}(\cdot)$, and ϵ. Note that, for a graph G, when $p \coloneqq \mathbf{mod}_{\mathcal{H}}(G) = 0$, and in this case our algorithm is essentially an EPTAS. In this sense, we generalize the assumed EPTAS on $\mathcal{H}$ to a larger family of graphs that are at most p vertices away from $\mathcal{H}$.

Let $M \subseteq V(G)$ be a modulator to $\mathcal{H}$ of size p, i.e., $G - M =: F \in \mathcal{H}$. We first compute a $(1 + \epsilon/2)$-approximate solution S for F using EPTAS in time $g(\epsilon) \cdot n^{\mathcal{O}(1)}$. Note that $S \subseteq V(G) \setminus M$, and $p \leq (1 + \epsilon/2) \cdot \mathsf{OPT}(F) \leq (1 + \epsilon/2)\mathsf{OPT}(G)$, where the last inequality follows from Observation 4. Depending on the relative sizes of M, and the approximate solution S, we consider the following two cases.

1. $p \leq \frac{\epsilon}{3} \cdot |S|$.
   Then we simply return $X \cup S$ as a solution for G. Note that $|M| + |S| \leq (1 + \epsilon/3)(1 + \epsilon/2) \cdot \mathsf{OPT}(F) \leq (1 + \epsilon) \cdot \mathsf{OPT}(F) \leq (1 + \epsilon) \cdot \mathsf{OPT}(G)$.
2. $p > \frac{\epsilon}{3} \cdot |S|$, i.e, $\mathsf{OPT}(F) \leq |S| \leq \frac{3p}{\epsilon}$.
   Then, by Observation 4, $\mathsf{OPT}(G) \leq \mathsf{OPT}(F) + |M| \leq p + \frac{3p}{\epsilon} =: p'$. In this case, we use an FPT algorithm parameterized by the solution size, to find an optimal solution for G. Note that this takes $h(p') \cdot n^{\mathcal{O}(1)}$, i.e., $f(p, \epsilon) \cdot n^{\mathcal{O}(1)}$ time, where $f(p, \epsilon) = h(p + 3p/\epsilon)$.

Thus, we get the following theorem.

▶ **Theorem 5.** *Let $\mathcal{F}, \mathcal{H}$ be well-behaved families of graphs. Moreover, suppose that* $\Pi =$ Vertex Deletion to $\mathcal{F}$ *admits an* EPTAS *on any graph in $\mathcal{H}$, and that Π is* FPT *parameterized by the solution size. Then, there exists an* FPT-AS *for Π, parameterized by $p \coloneqq \mathbf{mod}_{\mathcal{H}}(G)$ with running time* $\max\{g(\epsilon), f(p, \epsilon)\} \cdot n^{\mathcal{O}(1)}$.

Corollaries

Let Π be either Vertex Cover or Feedback Vertex Set. Note that Π is Vertex Deletion to $\mathcal{F}$, where $\mathcal{F}$ is family of isolated vertices, and forests, respectively. If $\mathcal{H}$ is a family of apex-minor free graphs, then Π admits an EPTAS on $\mathcal{H}$ with running time $2^{\mathcal{O}(1/\epsilon)} \cdot n^{\mathcal{O}(1)}$ ([7], Corollary 2). Furthermore, Π is FPT parameterized by solution size on general graphs, with running time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, where k denotes the solution size. Therefore, we get an FPT-AS parameterized by $p \coloneqq \mathbf{mod}_{\mathcal{H}}(G)$ with running time $2^{\mathcal{O}(p/\epsilon)} \cdot n^{\mathcal{O}(1)}$ for Vertex Cover and Feedback Vertex Set.

We note that it is possible to improve the running time of FPT-AS for Vertex Cover using an even simpler algorithm – we simply guess the intersection of an optimal solution with X by iterating over all subsets of X. Let $Y \subseteq X$ be a guess. If Y is feasible, then it must be that $U \coloneqq X \setminus Y$ must be an independent set, and $M \coloneqq N(U) \cap (V(G) \setminus X)$ must belong to the solution. Thus, the remaining graph is $G \setminus (M \cup X)$, which belongs to the family $\mathcal{H}$. Then, we use the $2^{\mathcal{O}(1/\epsilon)} \cdot n^{\mathcal{O}(1)}$ time EPTAS on $\mathcal{H}$ to obtain a solution S and return the smallest possible solution of the form $Y \cup M \cup S$. Thus, we get a $2^{\mathcal{O}(\mathbf{mod}_{\mathcal{H}}(G)+1/\epsilon)} \cdot n^{\mathcal{O}(1)}$ time FPT-AS.

▶ **Corollary 6.** *There exists an FPT-AS for* Vertex Cover *(resp.* Feedback Vertex Set*) that runs in time $2^{\mathcal{O}(p+\frac{1}{\epsilon})} \cdot n^{\mathcal{O}(1)}$ (resp. $2^{\mathcal{O}(p/\epsilon)} \cdot n^{\mathcal{O}(1)}$), where $p = \mathbf{mod}_{\mathcal{H}}(G)$, and $\mathcal{H}$ is a family of apex-minor free graphs.*

More generally, this result extends for Planar $\mathcal{R}$-Deletion, which is a special case of Vertex Deletion to $\mathcal{F}$, where the family $\mathcal{F}$ is characterized by a set $\mathcal{R}$ of forbidden minors containing at least one planar graph. We note that Fomin et al. [7] give an EPTAS for this problem on H-minor free graphs. On the other hand, this problem is known to admit a single-exponential FPT algorithm parameterized by the solution size via the results from Fomin et al. [6]. Thus, we get the following corollary.

▶ **Corollary 7.** *There exists an* FPT-AS *for* Planar $\mathcal{R}$-Deletion*, that runs in time $f(p, \epsilon) \cdot n^{\mathcal{O}(1)}$, where $p = \mathbf{mod}_{\mathcal{H}}(G)$, and $\mathcal{H}$ is a family of H-minor free graphs, for some fixed graph H and some computable function f. In particular, this implies an* FPT-AS *for* Treewidth-η-Modulator.[1]

Finally, we mention Odd Cycle Transversal. Note that there exists a polynomial time $\frac{9}{4}$-approximation for Odd Cycle Transversal on planar graphs [11], and there exists a $3^k n^{\mathcal{O}(1)}$ time algorithm for the problem parameterized by the solution size [20]. Thus, by using the 9/4-approximation instead of an $(1 + \epsilon/2)$-approximation on $G \setminus X$, followed by a similar case analysis, we get the following result.

▶ **Corollary 8.** *There exists an algorithm that runs in time $2^{\mathcal{O}(p/\epsilon)} \cdot n^{\mathcal{O}(1)}$, where p is the size of planar vertex deletion set, to compute a $(\frac{9}{4} + \epsilon)$-approximation for* Odd Cycle Transversal.

[1] For a fixed $\eta \geq 0$, Treewidth-η-Modulator is the problem of deciding whether one can delete at most k vertices from a given graph G, such that the resulting graph has treewidth at most η.

3.3 Equivalence Theorem

In this section we obtain the following equivalence statement.

▶ **Theorem 1.** *Let $\mathcal{H}, \mathcal{F}$ be well-behaved families of graphs, where $\mathcal{F}$ is CMSO-definable. Suppose $\Pi =$ Vertex Deletion to $\mathcal{F}$ has* FII*. Then, the following statements are equivalent.*

1. Π *admits an* FPT-AS *parameterized by* $\mathbf{mod}_{\mathcal{H}}(\cdot)$ *and* ϵ.
2. Π *admits an* FPT-AS *parameterized by* $\mathbf{ed}_{\mathcal{H}}(\cdot)$ *and* ϵ.
3. Π *admits an* FPT-AS *parameterized by* $\mathbf{tw}_{\mathcal{H}}(\cdot)$ *and* ϵ.

Proof. To prove that the three items are equivalent, we first note that for any graph G and any well-behaved family $\mathcal{H}$, it holds that $\mathbf{mod}_{\mathcal{H}}(G) \geq \mathbf{ed}_{\mathcal{H}}(G) \geq \mathbf{tw}_{\mathcal{H}}(G)$. Therefore, $3 \Longrightarrow 2 \Longrightarrow 1$. Thus, we focus on proving $1 \Longrightarrow 3$. Let $\mathcal{P}_{\mathbf{mod}}$ denote the assumed FPT-AS for Π parameterized by $\mathbf{mod}_{\mathcal{H}}$ and ϵ. That is, for any graph G on n vertices, $\mathcal{P}_{\mathbf{mod}}$ runs in time $f(\mathbf{mod}_{\mathcal{H}}(G), \epsilon) \cdot n^{\mathcal{O}(1)}$, and returns a subset $S \subseteq V(G)$ such that (i) $G - S \in \mathcal{F}$, and (ii) $|S| \leq (1+\epsilon) \cdot \mathsf{OPT}_{\Pi}(G)$. [2].

Let $G = (V, E)$ be a graph, and let (T, χ, L) be the given $\mathcal{H}$-tree decomposition of G of width at most $\mathbf{tw}_{\mathcal{H}}(G)$, and let $\ell \coloneqq \mathbf{tw}_{\mathcal{H}}(G)$ (resp. $\ell \coloneqq \mathbf{ed}_{\mathcal{H}}(G)$). Recall that $\mathcal{P}_{\mathbf{mod}}$ returns a $(1+\epsilon)$-approximation to Π on any instance (G, k) of Π.

For every leaf node $t \in V(T)$, let $V_t \coloneqq \chi(t)$, and let $H_t = V_t \cap L$ and $R_t = V_t \setminus L$. The properties of $\mathcal{H}$-tree decomposition imply that $G[H_t] \in \mathcal{H}$, and $|R_t| \leq \ell$. Equivalently, $R_t \subseteq V(G) \setminus L$ is a modulator to $\mathcal{H}$ for the graph $G[\chi(t)]$. Note that the base vertices, i.e., the vertices in L, are partitioned across the leaf bags, which implies that the graphs $\{G[H_t]\}_{\text{leaf } t \in V(T)}$ are disjoint.

We iterate over every leaf nodes $t \in V(T)$, and proceed as follows. Note that $G[H_t] \in \mathcal{H}$, thus $G[H_t]$ has a modulator of size zero to the family $\mathcal{H}$. Therefore, we can use $\mathcal{P}_{\mathbf{mod}}$ to obtain a set $S_t \subseteq H_t$ that is a $(1+\epsilon/2)$-approximation for Π on the graph $G[H_t]$, i.e., $|S_t| \leq (1+\epsilon/2) \cdot \mathsf{OPT}(G[H_t])$. This takes $f(0, \epsilon/2) \cdot |H_t|^{\mathcal{O}(1)} \leq h(\epsilon) \cdot n^{\mathcal{O}(1)}$ time for some function $h(\cdot)$. In other words, $\mathcal{P}_{\mathbf{mod}}$ is an EPTAS for Π on $G[H_t]$.

For every leaf node $t \in V(T)$, if $|R_t| \leq \epsilon/3 \cdot |S_t|$, then we say that the node is *good*. Otherwise, we say that $t \in V(T)$ is *bad*. Furthermore, all non-leaf nodes are classified as *bad*. Note that for a good node $t \in V(T)$, $|R_t| \leq (1+\epsilon/2) \cdot (\epsilon/3) \cdot \mathsf{OPT}(G[H_t]) \leq (\epsilon/2) \cdot \mathsf{OPT}(G[H_t])$. On the other hand, for a bad node $t \in V(T)$, $\ell \geq |R_t| \geq \epsilon/3 \cdot |S_t| \geq \epsilon/3 \cdot \mathsf{OPT}(G[H_t])$. Let U_g and U_b denote the set of good and bad nodes respectively. Let $S_1 = \bigcup_{t \in U_g} S_t$, and $S_2 = \bigcup_{t \in U_g} R_t$.

Furthermore, let $V_g = \bigcup_{t \in U_g} \chi(t)$, and $V_b = V \setminus V_g$. Now, let $S_b \subseteq V_b$ denote an $(1+\epsilon)$-approximate solution to Π on the graph $G[V_b]$. We first state the following lemma, whose proof is given after the current proof.

▶ **Lemma 9.** $|S| \leq (1+\epsilon) \cdot \mathsf{OPT}(G)$*, where* $S = S_1 \cup S_2 \cup S_b$ *as defined above.*

Let $F = G[V_b]$. From the $\mathcal{H}$-tree decomposition (T, χ, L) of G of width ℓ, it is possible to obtain a $\mathcal{H}$-tree decomposition of F of width at most ℓ, by deleting the vertices of V_g from every bag. For simplicity, we will continue to use (T, χ, L) for the new "projected" $\mathcal{H}$-tree decomposition of F. As suggested by Lemma 9, we need to find a set S_b, which is an $(1+\epsilon)$-approximation to Π on $F \coloneqq G[V_b]$. In fact, we will show how to find an optimal solution. To this end, we rely on the following technical lemma.

[2] Note that $\mathcal{P}_{\mathbf{mod}}$ may in fact be a family of approximation algorithms, containing an algorithm for each value of $\mathbf{mod}_{\mathcal{H}}(\cdot)$ and ϵ. We slightly abuse the notation and assume that we use an appropriate algorithm from this family, based on the value of $\mathbf{mod}_{\mathcal{H}}$ of the relevant graph, and ϵ.

▶ **Lemma 10.** *Let $\mathcal{H}, \mathcal{F}$ be well-behaved families of graphs, where $\mathcal{F}$ is CMSO-definable. Suppose that $\Pi =$ Vertex Deletion to $\mathcal{F}$ has* FII*. Furthermore, suppose there exists an* FPT-AS $\mathcal{P}_{\mathbf{mod}}$ *for Π, parameterized by $\mathbf{mod}_{\mathcal{H}}(\cdot)$ and ϵ. Then, for every positive integer ℓ, there exists an algorithm $\mathcal{A}_\ell$, that takes input (G, T, χ, L), where*

1. *G is a graph on at most n vertices, and*
2. *A $\mathcal{H}$-tree decomposition (T, χ, L) of G of width ℓ, such that:*
   - *$\star$ for every leaf $t \in V(T)$, the graph $H_t := G[\chi(t) \cap L]$ satisfies $\mathsf{OPT}_\Pi(H_t) \leq 3\ell/\epsilon$.*

The algorithm $\mathcal{A}_\ell$ runs in time $g(\ell, \epsilon) \cdot n^{\mathcal{O}(1)}$, where g is some function, and returns an optimal solution $S \subseteq V(G)$ to Π on G, i.e., $|S| = \mathsf{OPT}_\Pi(G)$, and $G - S \in \mathcal{F}$.

The idea of the proof of Lemma 10 is similar to that of Theorem 6.1 from [1]. However, they require that $\mathcal{P}_{\mathbf{mod}}$ be an *exact algorithm* for Π. Our result builds upon the fact that, when the input graph satisfies the property $\star$, an *approximation algorithm* can be made to behave like an exact algorithm by a suitable choice of ϵ. However, a formal proof requires several technical definitions pertaining to FII, and thus is deferred to the full version of the paper.

Now we finish the proof of Theorem 1. Note that the graph $G[V_b]$ and the $\mathcal{H}$-tree decomposition (T, χ, L) restricted to $G[V_b]$ satisfies the requirements of Lemma 10 by construction (recall that $V_b =$. Thus, using the algorithm from Lemma 10, we can, in fact, get an optimal solution $S_b \subseteq V_b$ to Π on $G[V_b]$. This concludes the proof of Theorem 1, modulo the proof of Lemma 9. ◀

Proof of Lemma 9. Recall the definitions of the partial solutions S_1, S_2, S_b as defined in the proof of Theorem 1. From Observation 4, we have the following inequality.

$$\mathsf{OPT}(G - S_2) \leq \mathsf{OPT}(G) \tag{1}$$

Note that the properties of $\mathcal{H}$-tree decomposition imply that, if we delete S_2 from G, the graph $G - S_2$ consists of disjoint induced subgraphs H_t for $t \in U_g$, as well as $G[V_b]$ (note that each of these induced subgraphs may or may not be connected). Now, consider

$$\begin{aligned}
|S| &= |S_1| + |S_2| + |S_b| && \text{(Since } S = S_1 \uplus S_2 \uplus S_b.\text{)}\\
&\leq \sum_{t \in U_g} (|S_t| + |R_t|) + |S_b| && \text{(Since } S_1 = \textstyle\bigcup_{t \in U_g} S_t \text{ and } S_2 = \bigcup_{t \in U_g} R_t\text{)}\\
&\leq \left(\sum_{t \in U_g} (1 + \epsilon/2) \cdot \mathsf{OPT}(G[H_t]) + \epsilon/2 \cdot \mathsf{OPT}(G[H_t]) \right) + (1 + \epsilon) \cdot \mathsf{OPT}(G[V_b])\\
& && \text{(Using bounds on } S_t, R_t \text{ and } S_b \text{ respectively)}\\
&\leq (1 + \epsilon) \cdot \left(\left(\sum_{t \in U_g} \mathsf{OPT}(G[H_t]) \right) + \mathsf{OPT}(G[V_b]) \right)\\
&\leq (1 + \epsilon) \cdot \mathsf{OPT}(G) && \text{(since } V(G) \setminus S_2 = V_b \uplus \biguplus_{t \in U_g} H_t \text{, and (1))}
\end{aligned}$$

◀

3.4 Variations of the Equivalence Theorem

Theorem 1 gives non-uniform FPT-ASes. In the following, we obtain a simpler, and *explicit* version of Theorem 1 for Vertex Deletion to $\mathcal{F}$ when $\mathcal{F}$ satisfies the following property, called *treewidth-η-modulated*: Suppose there exists a constant η, such that the (standard)

treewidth of every graph in $\mathcal{F}$ is bounded by η. In this case, we say that the family $\mathcal{F}$ is *treewidth-η-modulated*, or simply, *η-modulated*. Note that many natural Vertex Deletion to $\mathcal{F}$ problems are η-modulated. For example, Vertex Cover (resp. Feedback Vertex Set) is η-modulated for $\eta = 0$ (resp. $\eta = 1$). More generally, it is also known if the family $\mathcal{F}$ is characterized by a family of excluded minors $\mathcal{O}$ containing at least one planar graph, then Vertex Deletion to $\mathcal{F}$ is known as Planar $\mathcal{O}$-Deletion, and is also η-modulated for some η that depends on $\mathcal{O}$. The proof of this theorem is straightforward given the η-modulated property of $\mathcal{F}$. A formal proof can be found in the full version.

▶ **Theorem 11.** *Let $\mathcal{H}, \mathcal{F}$ be well-behaved families of graphs. Moreover, suppose that Π = Vertex Deletion to $\mathcal{F}$ is such that: (i) $\mathcal{F}$ is η-modulated for some constant $\eta \geq 0$, and (ii) there exists an exact* FPT *algorithm for Π parameterized by $\mathbf{tw}(\cdot)$. Then, the following statements are equivalent.*

1. *Π admits an* FPT-AS *parameterized by $\mathbf{mod}_{\mathcal{H}}(\cdot)$ and ϵ.*
2. *Π admits an* FPT-AS *parameterized by $\mathbf{ed}_{\mathcal{H}}(\cdot)$ and ϵ.*
3. *Π admits an* FPT-AS *parameterized by $\mathbf{tw}_{\mathcal{H}}(\cdot)$ and ϵ.*

Note that Theorem 11 requires that Π admit an FPT-AS parameterized by $\mathbf{mod}_{\mathcal{H}}(G)$. In particular, when $\mathbf{mod}_{\mathcal{H}}(G) = 0 \Longleftrightarrow G \in \mathcal{H}$, Π should have an EPTAS. However, in some cases, we only know an α-approximation for Π even when $G \in \mathcal{H}$, where $\alpha \geq 1$. In such cases, the following theorem may be applicable, if, Π is also known to be FPT parameterized by the size of the solution.

▶ **Theorem 12.** *Let $\mathcal{H}, \mathcal{F}$ be well-behaved families of graphs. Moreover, suppose that Π = Vertex Deletion to $\mathcal{F}$ satisfies the following properties: (i) Π is CMSO-definable, (ii) has* FII*, and (iii) Π is* FPT *parameterized by the size of the solution. Also suppose that for some constant $\alpha \geq 1$, Π admits an α-approximation in time $f(\mathbf{mod}_{\mathcal{H}}(\cdot)) \cdot n^{\mathcal{O}(1)}$. Then,*

1. *Π admits an $(\alpha + \epsilon)$-approximation in time $g(\mathbf{ed}_{\mathcal{H}}(\cdot), \epsilon) \cdot n^{\mathcal{O}(1)}$, and*
2. *Π admits an $(\alpha + \epsilon)$-approximation in time $h(\mathbf{tw}_{\mathcal{H}}(\cdot), \epsilon) \cdot n^{\mathcal{O}(1)}$.*

Plugging in Corollary 8 into Theorem 12, we obtain the following corollary.

▶ **Corollary 13.** *There exists an algorithm that runs in time $f(\ell, \epsilon) \cdot n^{\mathcal{O}(1)}$, where ℓ is $\mathbf{ed}_{\mathcal{H}}(G)$ (or $\mathbf{tw}_{\mathcal{H}}(G)$) to planar graphs, and computes a $(\frac{9}{4} + \epsilon)$-approximation for* Odd Cycle Transversal.

4 FPT-ASes for Dominating Set

In this section, we consider Dominating Set, defined as follows.

> Blue-White Dominating Set
> **Input:** An instance (G, k), where G is a graph, and k is a non-negative integer
> **Question:** Does G contain a dominating set of size at most k, i.e., does there exist a set $S \subseteq V(G)$ such that for each $u \in V(G)$, $N[v] \cap S \neq \emptyset$?

Note that the main difficulty is that Dominating Set is not monotone, i.e., it does not necessarily hold that for any $S \subseteq V(G)$, $\mathsf{OPT}(G - S) \leq \mathsf{OPT}(G)$. Thus, the approach for Vertex Deletion to $\mathcal{F}$ does not immediately generalize, and the arguments for Dominating Set are technically more involved. Furthermore, it is not immediately clear that Dominating Set is *self-reducible*. Therefore, we need to rely on different *annotated versions* of Dominating Set to design approximation algorithms for the problem.

First, the following theorem that is proved using arguments involving bidimensionality. The formal proof of the theorem can be found in the full version in the appendix.

▶ **Theorem 14.** *Suppose we are given a graph G, and a set $M \subseteq V(G)$ of size p, such that $G' := G \setminus M \in \mathcal{H}$, where $\mathcal{H}$ is a family of apex-minor free graphs. Then, there exists an $f(p,\epsilon) \cdot n^{\mathcal{O}(1)}$ time algorithm to find a $(1+\epsilon)$-approximation for* Dominating Set *on G, where $f(p,\epsilon) = \max\left\{2^{\mathcal{O}(p+\sqrt{\frac{p}{\epsilon}})},\ \ 2^{\mathcal{O}(1/\epsilon)}\right\}$.*

Next, we prove the equivalence theorem (cf. Theorem 15), which implies FPT-AS for Dominating Set, parameterized by $\mathbf{ed}_{\mathcal{H}}(\cdot)$ (resp. $\mathbf{tw}_{\mathcal{H}}(\cdot)$) and ϵ. The proof of this theorem is conceptually similar to that of Theorem 1; however, it is technically much more involved due to the reasons mentioned above.

4.1 Equivalence Theorem for Dominating Set

In this section, we prove the following equivalence theorem for Dominating Set, assuming that $\mathcal{H}$ be *well-behaved*, i.e., hereditary and closed under disjoint union.

▶ **Theorem 15.** *Let $\mathcal{H}$ be any well-behaved family of graphs, and let $\Pi =$ Dominating Set. Then, the following statements are equivalent.*

1. *Π admits an* FPT-AS *parameterized by $\mathbf{mod}_{\mathcal{H}}(\cdot)$ and ϵ.*
2. *Π admits an* FPT-AS *parameterized by $\mathbf{ed}_{\mathcal{H}}(\cdot)$ and ϵ.*
3. *Π admits an* FPT-AS *parameterized by $\mathbf{tw}_{\mathcal{H}}(\cdot)$ and ϵ.*

Proof. We prove 1$\Longrightarrow$3. Let $\mathcal{P}_{\mathbf{mod}}$ be the assumed FPT-AS for Π parameterized by $\mathbf{mod}_{\mathcal{H}}$ and ϵ. Let $G = (V, E)$ be the input graph on n vertices, along with an $\mathcal{H}$-tree decomposition (T, χ, L). Let ℓ denote the width of the given $\mathcal{H}$-tree decomposition. First, if $\mathcal{H}$ only contains the empty graph $(\emptyset, \emptyset)$, our goal is to prove that Dominating Set admits an FPT-AS parameterized by (standard) $\mathbf{tw}(\cdot)$. Indeed, since Dominating Set is FPT parameterized by $\mathbf{tw}(\cdot)$, we assume that $\mathcal{H}$ contains at least one non-empty graph. In particular, due to hereditary property, $\mathcal{H}$ must contain a graph on a single vertex. Furthermore, we assume that $\ell \geq 1$ – otherwise G is a disjoint union of graphs belonging to $\mathcal{H}$. This implies that $G \in \mathcal{H}$, in which case all three statements are trivially equivalent.

As before, for any leaf node $t \in V(T)$, let $V_t = \chi(t)$, $H_t = V_t \cap L$, and $R_t = V_t \setminus H_t$. Note that $G[H_t] \in \mathcal{H}$, and the sets $\{H_t\}_{t \in V(T)}$ are pairwise disjoint, and $|R_t| \leq \ell$ for any $t \in V(T)$.

We iterate over every leaf node $t \in V(T)$, and create a new graph $\widetilde{G}_t$ as follows. Let v^* be a new vertex that is not in V_t. Let $V'_t = V_t \cup \{v^*\}$. Let $E'_t = E(G[V_t]) \cup \widetilde{E}$, where $\widetilde{E} = \{v^*u : u \in R_t\}$. That is, the graph $\widetilde{G}_t$ is obtained by adding a new vertex v^* to the graph $G[V_t]$ and making it adjacent to all vertices of R_t.

Note that $\widetilde{G}_t - R_t = G[H_t] \uplus G^*$, where $G^* = (v^*, \emptyset)$ is an isolated component containing v^*. Now, $G[H_t] \in \mathcal{H}$, and $G^* \in \mathcal{H}$, which implies that $\widetilde{G}_t - R_t = G[H_t] \uplus G^* \in \mathcal{H}$, since $\mathcal{H}$ is closed under disjoint union. Now, we use $\mathcal{P}_{\mathbf{mod}}$ on $\widetilde{G}_t$ to obtain $\widetilde{S}_t \subseteq V(\widetilde{G}_t)$, such that (i) $\widetilde{S}_t$ is a dominating set for $\widetilde{G}_t$, and (ii) $|\widetilde{S}_t| \leq (1+\epsilon/4) \cdot \mathsf{OPT}(\widetilde{G}_t)$. Let $S_t = \widetilde{S}_t \setminus \{v^*\}$.

For a leaf node $t \in V(T)$ if $\ell \leq \frac{\epsilon}{29} \cdot |S_t|$, then we say that t is a *good* node; and *bad* otherwise. All internal nodes are classified as bad. Let U_g, U_b denote the sets of good and bad nodes respectively. Let $V_g = \bigcup_{t \in U_g} H_t$, $V_m = \bigcup_{t \in U_g} R_t$, and $V_b = V(G) \setminus (V_g \cup V_m)$.

Let D^* be an optimal dominating set for G, and for any $t \in U_g$, let $D^*_{1,t} = H_t \cap D^*$, and $D^*_{2,t} = R_t \cap D^*$, and $D^*_t = D^*_{1,t} \cup D^*_{2,t}$. Let $D^*_1 = \bigcup_{t \in U_G} D^*_{1,t}$, and $D^*_2 = \bigcup_{t \in U_g} D^*_{2,t}$. Note that $\left\{D^*_{1,t}\right\}_{t \in U_g}$ is a partition of D^*_1, whereas a vertex in D^*_2 may belong to multiple $D^*_{2,t}$'s. Finally, let $D^*_b = D^* \cap V_b$. Note that $\{D^*_1, D^*_2, D^*_b\}$ is a partition of D^*.

Now, let $S_1 := \bigcup_{t \in U_g} S_t$, and $S_2 := \bigcup_{t \in U_g} R_t$. We wish to add $S_1 \cup S_2$ to our solution. Since each R_t is a separator between H_t and $G - \chi(t)$, we can delete V_g from the graph. However, we cannot simply delete $V_m = S_2$ from the graph – since the vertices in S_2 may already dominate a subset of vertices in V_b. Therefore, we need to "remember" that we have decided to add S_2 into our solution. To this end, we define an *annotated* version of Dominating Set, which we call Blue-White Dominating Set. The Blue-White Dominating Set problem is defined as follows.

Blue-White Dominating Set
Input: An instance (G, B, W, k), where G is a graph, $V(G) = B \uplus W$, and k is a non-negative integer
Question: Does there exist a subset $S \subseteq V(G)$ of size at most k such that (i) S is a dominating set for G, and (ii) $B \subseteq S$?

Let $F := G[S_2 \uplus V_b]$. We want to find the smallest integer k', such that (F, S_2, V_b, k') is a yes-instance of Blue-White Dominating Set. Let S_b denote an $(1 + \epsilon/2)$-approximate solution on this instance, i.e., (i) $S_b \subseteq S_2 \cup V_b$ is a dominating set for F, (ii) $S_2 \subseteq S_b$, and (iii) $|S_b| \leq (1 + \epsilon/2) \cdot \mathsf{OPT}'(F)$, where $\mathsf{OPT}'(F)$ denotes the size of the optimal solution for the Blue-White Dominating Set instance. We prove the following technical lemma, the proof of which is rather involved, and thus given in the full version.

▶ **Lemma 16.** $|S| = |S_1| + |S_2| + |S_b| \leq (1 + \epsilon) \cdot \mathsf{OPT}(G)$

Thus, our task is reduced to finding an approximate solution to Blue-White Dominating Set on (F, S_2, V_b, k'); in fact, we will find an exact solution. To do this, first we prove the following lemma, which shows that Blue-White Dominating Set can be reduced to the (standard) dominating problem on an auxiliary graph. The proof is straightforward, and is given in the full version.

▶ **Lemma 17.** *Let $\mathcal{I} = (G, B, W, k)$ be an instance of* Blue-White Dominating Set. *Let G' be the graph obtained by attaching $N = n^2$ distinct pendant vertices to each vertex in B, where $n = |V(G)|$. Let $\mathcal{I}' = (G', k)$ be the resulting instance of* Dominating Set. *Then $\mathcal{I}$ is a yes-instance of* Blue-White Dominating Set *iff $\mathcal{I}'$ is a yes-instance of* Dominating Set.

As suggested by Lemma 17, we create a graph F', as follows. For each $u \in S_2$, let P_u denote the set of $|V(F)|^2$ distinct pendant vertices attached to u. By slightly abusing the notation, let (T, χ, L) be the original $\mathcal{H}$-tree decomposition, restricted to the vertices of F. We modify this to obtain an $\mathcal{H}$-tree decomposition for F'. Consider a vertex $u \in S_2$, and note that $u \in R_t$ for some $t \in V(T)$. We pick one such arbitrary node t, and add a child t_u, which becomes a leaf in T. For this node t_u, we define its bag $\chi'(t_u) = P_u \cup \{u\}$, where P_u becomes the set of *base vertices* in $\chi'(t_u)$, and u is a non-base vertex in $\chi'(t_u)$. Note that $G[P_u]$ is a set of isolated vertices, and thus belongs to $\mathcal{H}$. For all original nodes $t \in V(T)$, we set $\chi'(t) = \chi(t)$. Finally, $L' = \bigcup_{u \in S_2} P_u$, and let T' be the resulting tree. It is easy to see that (T', χ', L') is a valid $\mathcal{H}$-tree decomposition for F' of width at most ℓ.

Now, we can prove an analogous version of Lemma 10 for the instance (F, k') of Dominating Set. It is known that Dominating Set is CMSO-definable and has FII [8]. We highlight the key differences required to perform replacement using the FII property of Dominating Set. Note that for a node that was earlier classified as *bad*, $\mathsf{OPT}(F[H_t]) = \mathsf{OPT}(G[H_t]) \leq 29\ell/\epsilon$, which implies that $\mathsf{OPT}(F[\chi'(t)]) \leq 29\ell/\epsilon + \ell \leq 30\ell/\epsilon$. On the other hand, if t is a

newly added leaf node then $\mathsf{OPT}(F[\chi'(t)]) \leq |R_t| \leq \ell$. This implies that, when we want to find a replacement for $G[\chi'(t)]$ using an application of FII, we want to decide instances of the form $F[\chi'(t)] \oplus Y$, where $|V(Y)|$ is upper bounded by some function of ℓ. Therefore, $\mathsf{OPT}(F[\chi'(t)] \oplus Y) \leq \mathsf{OPT}(F[\chi'(t)]) + |V(Y)|$, which is upper bounded by some function of ℓ and ϵ. Therefore, a version of Lemma 10 can solve *the decision version* of Blue-White Dominating Set.

Now, we discuss how to find a solution, using the algorithm for the decision version as an oracle. We try the following for the values of $k = |S_2|, |S_2| + 1, \ldots$, until the first time the decision version returns that Blue-White Dominating Set$(F, S_2, V(F) \setminus S_2, k)$ is a yes-instance. Let k^* denote this value.

We maintain a partial solution S_b, which is initialized to S_2. Consider a leaf node $t \in V(T)$, such that S_b does not already dominate all vertices of $\chi'(t) \cap L'$. We pick an arbitrary *original* vertex $u \in \chi'(t) \cap L'$, and use the algorithm to decide the instance $(F, S_b \cup \{u\}, V(F) \setminus (S_b \cup \{u\}), k^*)$ of Blue-White Dominating Set. If the algorithm returns no, then we conclude that u does not belong to a solution of size k^*. Otherwise, we update S_b by adding u to it, and proceed to the next iteration.

Now we discuss the technicalities when we use the reduction from Lemma 17 on the newly created Blue-White Dominating Set instance. In this reduction, we add $N = n^2$ distinct pendant vertices to u, and let F_u be the resulting graph. We add all these vertices to L' and to $\chi'(t)$. We also remove u from L, i.e., move it from H_t to R_t. Note that the resulting base graph is a disjoint union of $F'[H_t] - u$ and the N pendant vertices, each of which belong to $\mathcal{H}$ due to the well-behaved property. The size of the modulator increases by 1. However, since $\mathsf{OPT}(F[\chi(t)])$ is bounded by $29\ell/\epsilon$, therefore, we move at most $29\ell/\epsilon$ vertices from H_t to L_t. Thus, the width of the resulting $\mathcal{H}$-tree decomposition remains bounded by $\ell + 29\ell/\epsilon$. Furthermore, even for an incorrect choice of u, $\mathsf{OPT}(F'[\chi'(t)])$ remains bounded by a function of ℓ and ϵ. Thus, a version of Lemma 10 can still be used to obtain a solution S_b of size k^* to the original instance (F, S_2, V_b, k^*). This concludes the proof of the theorem. ◀

5 Conclusion

In this paper, we have initiated a systematic exploration of the impact that recently introduced "hybrid" graph parameters have on the existence of good approximations for fundamental graph problems. In fact, we have shown that as far as the task of obtaining an FPT-AS is concerned, for many problems, designing an FPT-AS parameterized by the largest of these, i.e., $\mathbf{mod}_{\mathcal{H}}$, is sufficient to obtain an FPT-AS parameterized by both $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$. This result gives an approximation analogue of recent equivalence obtained between these parameters in the exact algorithmic setting.

To demonstrate concrete applicability of our techniques, we first designed FPT-ASes for many classical graph problems parameterized by $\mathbf{mod}_{\mathcal{H}}$, where $\mathcal{H}$ is an apex/H-minor free graph family. Then, using our equivalence theorems, we are able to lift these FPT-ASes to $\mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$. At this point, we would like to highlight that, in several concrete applications of our equivalence theorems, the *non-uniform* FPT-ASes can be made uniform. For example, we believe that one can obtain uniform FPT-ASes parameterized by $\mathbf{mod}_{\mathcal{H}}, \mathbf{ed}_{\mathcal{H}}$ and $\mathbf{tw}_{\mathcal{H}}$ for problems such as Vertex Cover, Feedback Vertex Set, Independent Set, Dominating Set, Cycle Packing, where $\mathcal{H}$ is an apex-minor free graph family.

We conclude with two final remarks. Firstly, the assumption of CMSO-definability can be relaxed. In fact, we only require an EPTAS for the problem on graphs of bounded treewidth. This would then enable one to apply our framework to problems that are not

CMSO-definable but are known to have good approximations on the graph family under consideration. Secondly, the initial step of our algorithm, i.e., FPT-ASes parameterized by $\mathbf{mod}_{\mathcal{H}}$, can be made to work with other graph families, such as (Unit) Disk Graphs, using known EPTASes for a number of graph problems [7, 17].

References

1. Akanksha Agrawal, Lawqueen Kanesh, Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Deleting, eliminating and decomposing to hereditary classes are all FPT-equivalent. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1976–2004, 2022. doi:10.1137/1.9781611977073.79.
2. Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016. doi:10.1007/s00453-015-0045-3.
3. Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
4. Erik D. Demaine, Timothy D. Goodrich, Kyle Kloster, Brian Lavallee, Quanquan C. Liu, Blair D. Sullivan, Ali Vakilian, and Andrew van der Poel. Structural rounding: Approximation algorithms for graphs near an algorithmically tractable class. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.37.
5. Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/a13060146.
6. 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.
7. Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded grid minors and efficient polynomial-time approximation schemes. *J. ACM*, 65(2):10:1–10:44, 2018. doi:10.1145/3154833.
8. Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.
9. Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining treewidth and backdoors for CSP. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 36:1–36:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.36.
10. M. R. Garey and David S. Johnson. The rectilinear steiner tree problem is NP complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.
11. Michel X. Goemans and David P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Comb.*, 18(1):37–59, 1998. doi:10.1007/PL00009810.
12. Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2004. doi:10.1007/978-3-540-28639-4_15.
13. Tanmay Inamdar, Lawqueen Kanesh, Madhumita Kundu, M. S. Ramanujan, and Saket Saurabh. Fpt approximations for packing and covering problems parameterized by elimination distance and even less, 2023. arXiv:2310.03469.

14 Bart M. P. Jansen and Jari J. H. de Kroon. FPT algorithms to compute the elimination distance to bipartite graphs and more. In Lukasz Kowalik, Michal Pilipczuk, and Pawel Rzazewski, editors, *Graph-Theoretic Concepts in Computer Science – 47th International Workshop, WG 2021, Warsaw, Poland, June 23-25, 2021, Revised Selected Papers*, volume 12911 of *Lecture Notes in Computer Science*, pages 80–93. Springer, 2021. doi:10.1007/978-3-030-86838-3_6.

15 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. 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 1757–1769. ACM, 2021. doi:10.1145/3406325.3451068.

16 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. 5-approximation for $\mathcal{H}$-treewidth essentially as fast as $\mathcal{H}$-deletion parameterized by solution size. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 66:1–66:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.66.

17 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A framework for approximation schemes on disk graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2228–2241. SIAM, 2023. doi:10.1137/1.9781611977554.ch84.

18 Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008. doi:10.1093/comjnl/bxm048.

19 Laure Morelle, Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. Faster parameterized algorithms for modification problems to minor-closed classes. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 93:1–93:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.93.

20 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.

Leakage Resilience, Targeted Pseudorandom Generators, and Mild Derandomization of Arthur-Merlin Protocols

Dieter van Melkebeek
University of Wisconsin-Madison, WI, USA

Nicollas Mocelin Sdroievski
University of Wisconsin-Madison, WI, USA

Abstract

Many derandomization results for probabilistic decision processes have been ported to the setting of Arthur-Merlin protocols. Whereas the ultimate goal in the first setting consists of efficient simulations on deterministic machines (BPP vs. P problem), in the second setting it is efficient simulations on nondeterministic machines (AM vs. NP problem). Two notable exceptions that have not yet been ported from the first to the second setting are the equivalence between whitebox derandomization and leakage resilience (Liu and Pass, 2023), and the equivalence between whitebox derandomization and targeted pseudorandom generators (Goldreich, 2011). We develop both equivalences for mild derandomizations of Arthur-Merlin protocols, i.e., simulations on Σ_2-machines. Our techniques also apply to natural simulation models that are intermediate between nondeterministic machines and Σ_2-machines.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Hardness versus randomness tradeoff, leakage resilience, Arthur-Merlin protocol, targeted hitting set generator

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.29

Funding Partial support for this research was provided by the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation, and by the National Science Foundation under Grant No. 2312540.

1 Introduction

Understanding whether bounded-error probabilistic algorithms can be simulated deterministically with a polynomial overhead in time (the BPP versus P problem) is a central problem in the theory of computing. An analogous question in relation to interactive protocols is whether constant-round interactive protocols (commonly referred to as Arthur-Merlin protocols), can be simulated *nondeterministically* with a polynomial overhead in time (the AM versus NP problem). An influential line of research on hardness vs. randomness tradeoffs concludes derandomization from lower bounds, and in some cases equivalences are known. As the lower bounds are plausible, it is conjectured that $\mathsf{BPP} = \mathsf{P}$ and $\mathsf{AM} = \mathsf{NP}$. Still, even *mild* derandomizations that require more nondeterminism than what is conjectured (such as $\mathsf{BPP} \subseteq \mathsf{NP}$ and $\mathsf{AM} \subseteq \Sigma_2\mathsf{P} = \mathsf{NP}^{\mathsf{NP}}$) remain open. We first describe the situation for the BPP setting and then move on to the AM setting, which is the focus of this work.

BPP setting. The first hardness vs. randomness tradeoffs for BPP were in the *blackbox* setting, where a pseudorandom generator (PRG) obliviously constructs a small set of strings that "fools" every efficient probabilistic algorithm in the sense that the algorithm behaves approximately the same under the uniform distribution and the generator's distribution. To construct pseudorandom generators, early works relied on non-uniform hardness assumptions

© Dieter van Melkebeek and Nicollas Mocelin Sdroievski;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 29; pp. 29:1–29:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

such as linear-exponential circuit lower-bounds for $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ (which suffices to conclude $\mathsf{BPP} = \mathsf{P}$) [12]. The preceding parameter setting is known as the high end of the derandomization spectrum. At the low end of the spectrum, subexponential-time blackbox simulations for BPP follow from $\mathsf{E} \not\subseteq \mathsf{P/poly}$, i.e., polynomial-size circuit lower bounds for linear-exponential time [3]. Pseudorandom generators are in fact *equivalent* to non-uniform lower bounds, and the equivalence holds across the entire derandomization spectrum [22].

The discussion presents an equivalence between *blackbox* derandomization and non-uniform lower bounds. It leaves the question of whether general (*whitebox*) derandomization implies non-uniform lower bounds. Given the equivalence between such lower bounds and PRGs, another way to phrase the question is whether PRGs are "complete" for derandomization: If derandomization is possible, then is it possible through PRGs?

Goldreich [8, 9] showed that whitebox derandomization of prBPP (the promise-problem version of BPP) is equivalent to the existence of *targeted* pseudorandom generators, which are pseudorandom generators that receive as input a target composed of an algorithm A and an input x, and are only required to "fool" A on input x. The equivalence establishes a "normal form" for derandomizing prBPP. In principle, a deterministic simulation A_{det} of a probabilistic algorithm A can access A in other ways than by running A on the given input x and some random bit strings ρ. Goldreich's result, however, establishes that if derandomization is possible, i.e., $\mathsf{prBPP} \subseteq \mathsf{P}$, then we can always take A_{det} to be the algorithm that, on input x, computes a targeted pseudorandom set $S_{A,x}$ and outputs the majority answer obtained by simulating A on input x while replacing its random inputs by each element in $S_{A,x}$.

Recently, Chen and Tell [5] proposed a candidate hardness assumption that is equivalent to whitebox derandomization of prBPP: *uniform* lower bounds for *multi-bit* functions f (rather than usual decision problems) against probabilistic algorithms that hold on *almost-all inputs*, i.e., any probabilistic algorithm limited to a certain running time can only compute f correctly on a finite number of inputs. Chen and Tell show that if there exists a length-preserving function f computable by logspace-uniform circuits of polynomial size and depth n^b for some constant b that is hard on almost-all inputs against $\mathsf{prBPTIME}[n^{b+O(1)}]$, then $\mathsf{prBPP} \subseteq \mathsf{P}$. They also observe that $\mathsf{prBPP} \subseteq \mathsf{P}$ implies that for every constant c, there is a length-preserving function $f \in \mathsf{P}$ that is hard on almost-all inputs against $\mathsf{prBPTIME}[n^c]$. Modulo the uniform depth requirement, their result establishes a near-equivalence between hardness on almost-all inputs and derandomization. The approach scales fairly well to the rest of the spectrum, but not perfectly. In particular, the low end remains open.

Under a different hardness assumption, Liu and Pass managed to obtain *full* equivalences for derandomization of prBPP and hardness on almost-all inputs. The hardness assumption that they employ is the existence of computational problems (relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$) that are hard even in the presence of efficiently-computable leakage [15]. We define a version of the notion in the setting of hardness on almost-all inputs.

▶ **Definition 1** (Leakage-resilient hardness). *A relation R is (T, ℓ)-leakage-resilient hard on almost-all inputs against a class of algorithms if for all pairs* (Leak, A) *of algorithms in the class that run in time T and such that $|\text{Leak}(x, y)| \leq \ell(|x|)$, the following holds for almost-all inputs x and every $y \in R(x) = \{y \mid (x, y) \in R\}$: The probability that $A(x, \text{Leak}(x, y)) = y$ is at most $1/3$.*

In cryptographic terms, the algorithm A can be viewed as an attacker that attempts to compute $y \in R(x)$ by receiving a few bits of information about y from the leakage-providing algorithm Leak. In some cases, such as when the relation R is computable in P, one may take R to be a function.

At the high end of the derandomization spectrum, the main result of Liu and Pass [15] establishes an equivalence between the derandomization $\mathsf{prBPP} \subseteq \mathsf{P}$ and the existence of a length-preserving function $f \in \mathsf{P}$ that is leakage-resilient hard on almost all inputs against probabilistic algorithms running in fixed-polynomial time and with sublinear leakage. The equivalence works across the entire derandomization spectrum. For completeness we mention that, in another work, Liu and Pass establish an equivalence between derandomization and hardness related to approximating conditional Levin-Kolmogorov complexity (Kt) [14]. The latter equivalence does not scale well toward the low end, though.

One can also ask about efficient simulations of prBPP on *nondeterministic* machines. Such simulations are referred to as *mild* derandomizations. We remark that establishing mild derandomization results for prBPP is a required step if we hope to show stronger deterministic simulations for the class. At the low-end of the derandomization spectrum, an equivalence is known between whitebox and blackbox mild derandomizations for prBPP that work for infinitely-many input lengths [11]. The technique, however, does not scale and is only known to work in the infinitely-often setting. The Liu-Pass result on the connection between leakage-resilient hardness and derandomization extends to the mild setting. In contrast to the low-end equivalence of [11], Theorem 2 extends to the entire derandomization spectrum. We state the high-end version.

▶ **Theorem 2** (Follows from [15, Theorem 1.6]). *There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prBPP} \subseteq \mathsf{NP}$.
- *There exists a total length-preserving relation $R \in \mathsf{NP}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against* prBPP.

AM setting. In the *blackbox* setting, an equivalence between derandomization and non-uniform lower bounds is known throughout the entire spectrum [13, 16, 18]. The class E is replaced by $\mathsf{NE} \cap \mathsf{coNE}$, and the circuits are nondeterministic (or single-valued nondeterministic, or deterministic with oracle access to an NP-complete problem like SAT). These assumptions yield hitting-set generators (HSGs) for AM, the one-sided error counterparts of pseudorandom generators. These objects suffice for derandomizing AM as every Arthur-Merlin protocol can be efficiently transformed into an equivalent one with perfect completeness [7].

Until recently, not much was known in the *whitebox* setting for AM. Van Melkebeek and Sdroievski have established a near-equivalence between (plain) almost-all inputs hardness and derandomization [23]. In one direction, they show that the existence of a length-preserving function computable in nondeterministic time n^b that is hard on almost-all inputs against prAM protocols that run in time $n^{O((\log b)^2)}$ implies that $\mathsf{prAM} \subseteq \mathsf{NP}$. They also establish a converse, though the hard function they obtain requires a small amount of advice and the hardness is only guaranteed against AM protocols instead of prAM protocols. The hardness-to-derandomization direction of their result scales very well toward the low-end.

It remains open whether whitebox derandomization of prAM can always be obtained through targeted hitting-set generators if it can be obtained at all, a question raised by Goldreich [8]. As a byproduct of their main result, the authors of [23] obtain a first step toward a positive resolution: They show that any low-end derandomization of prAM implies the existence of a targeted hitting-set generator that achieves a slightly weaker derandomization.

Similar to the BPP setting, one may ask for derandomizations of AM that require more resources than what is conjectured to suffice. While it is believed that $\mathsf{prAM} \subseteq \mathsf{NP}$ and we know that $\mathsf{prAM} \subseteq \Pi_2\mathsf{P} = \mathsf{coNP}^{\mathsf{NP}}$, it remains open whether $\mathsf{prAM} \subseteq \mathsf{P}^{\mathsf{NP}}$ or even whether $\mathsf{prAM} \subseteq \Sigma_2\mathsf{P} = \mathsf{NP}^{\mathsf{NP}}$, both of which are required steps toward showing that $\mathsf{prAM} \subseteq \mathsf{NP}$.

We note that even a subexponential-time Σ_2-simulation with subpolynomial advice for prAM that works for infinitely many input lengths remains open.[1] In this setting, an equivalence between whitebox and blackbox derandomization is known: The simulation is equivalent to polynomial size *nondeterministic* circuit lower bounds for the class $\Sigma_2\mathsf{E}$. In symbols, $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all ϵ if and only if $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ [2]. The equivalence does not scale well, and it is unknown whether it holds for other parameter settings such as the high end.

Our results. Our main contribution is establishing *full* equivalences for *mild* derandomization of prAM. Specifically, we obtain an equivalence between mild derandomization of prAM and leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, the class of polynomial-time probabilistic algorithms with non-adaptive oracle access to SAT. The equivalence scales smoothly across the entire derandomization spectrum, and applies to other classes between NP and $\Sigma_2\mathsf{P}$ in addition to $\Sigma_2\mathsf{P}$. We state the high-end version.

▶ **Theorem 3.** *Let $\mathcal{C} \in \{\mathsf{P}^{\mathsf{NP}}, \mathsf{ZPP}^{\mathsf{NP}}, \Sigma_2\mathsf{P}\}$. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prAM} \subseteq \mathcal{C}$.
- *There exists a total length-preserving relation $R \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

Theorem 3 can be viewed as a counterpart to Theorem 2 in the mild setting for prAM. Indeed, when compared to the traditional setting, both require an extra level of nondeterminism in the hardness assumption as well as in the simulation.

The class $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ was used in the initial derandomization results for Arthur-Merlin protocols [13] and is seemingly more powerful than prAM. However, derandomization results for prAM typically translate into corresponding derandomization results for $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. In particular, the derandomization $\mathsf{prAM} \subseteq \Sigma_2\mathsf{P}$ of Theorem 3 implies that $\mathsf{prBPP}_{||}^{\mathsf{SAT}} \subseteq \Sigma_2\mathsf{P}$. Also, lower bounds for linear-exponential time classes such as E and NE against nondeterministic circuits (which suffice for derandomizing prAM) imply non-uniform lower bounds for the same class against deterministic circuits with non-adaptive oracle access to SAT (which suffice for derandomizing $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$) [19].

We show that a connection between $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ and prAM holds in the leakage-resilient hardness setting as well. Specifically, we show that the derandomization conclusion of Theorem 3 holds under a weaker hardness assumption on *learn-and-evaluate* Arthur-Merlin protocols. Learn-and-evaluate protocols are a two-phase type of Arthur-Merlin protocol that arises naturally in hardness vs. randomness tradeoffs and fits nicely into the leakage-resilient hardness paradigm. In the first phase, a probabilistic algorithm with oracle access to a function f produces a short sketch π that consists of evaluations of f. In our case, f is the function that maps an index i to the i-th bit of a string y such that $(x, y) \in R$, and the first phase can be viewed as a small amount of leakage on y. In the second phase, an Arthur-Merlin protocol computes f given π, which naturally translates into a protocol that attempts to recover y from a "small" amount of leakage. The result, taken together with the derandomization-to-hardness direction of Theorem 3, implies an equivalence between leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ and learn-and-evaluate protocols.

[1] The best known unconditional results are worst-case simulations of AM in $\text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^c$ for all ϵ and some fixed c [24] and average-case simulations of prAM in $\text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all ϵ [23].

▶ **Theorem 4.** *Let $\mathcal{C} \in \{\mathsf{P}^{\mathsf{NP}}, \mathsf{ZPP}^{\mathsf{NP}}, \Sigma_2\mathsf{P}\}$. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

1. *There exists a total length-preserving relation $R \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*
2. *There exists a total length-preserving relation $R \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against learn-and-evaluate protocols.*

Theorem 4 partially addresses an open problem posed by Shaltiel and Umans of obtaining uniform equivalences between hardness against Arthur-Merlin protocols and algorithms with non-adaptive oracle access to SAT [19]. They ask whether an implication such as $\mathsf{E} \not\subseteq \mathsf{AM} \implies \mathsf{E} \not\subseteq \mathsf{P}_{||}^{\mathsf{SAT}}$ holds. Theorem 4 establishes an analogous result in the mild leakage-resilient hardness setting.

We also address, in the mild setting, the open problem posed by Goldreich about prAM [8]. We show that mild derandomization of prAM is equivalent to the existence of targeted hitting-set generators achieving the same derandomization. We state such a result at the high end of the derandomization spectrum, but it extends to the entire range.

▶ **Theorem 5.** *Let $\mathcal{C} \in \{\mathsf{P}^{\mathsf{NP}}, \mathsf{ZPP}^{\mathsf{NP}}, \Sigma_2\mathsf{P}\}$. If $\mathsf{prAM} \subseteq \mathcal{C}$, then there exist targeted hitting-set generators that achieve this mild derandomization result.*

Finally, we connect leakage-resilient hardness to the known equivalence between whitebox and blackbox mild derandomization of prAM at the low end. As mentioned before, the simulation $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$ is equivalent to the non-uniform lower bound $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ [2]. We show that those conditions are also equivalent to leakage-resilient hardness where the algorithm A needs to produce each individual bit of a solution $y \in R(x)$ very efficiently. The equivalence of [2] holds in the infinitely-often setting. Correspondingly, the equivalent leakage-resilient hardness condition holds for all inputs of infinitely-many input lengths.

▶ **Definition 6** (Local Leakage-resilient hardness)**.** *A relation R is t-local (T, ℓ)-leakage-resilient hard on all inputs of infinitely-many input lengths against a class of algorithms if for all pairs (Leak, A) of algorithms in the class such that Leak runs in time T, $|\mathrm{Leak}(x, f(x))| \leq \ell(|x|)$ and A runs in time t, the following holds for infinitely many $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$: $R(x)$ is non-empty and for every such $y \in R(x)$ there exists $i \in [|y|]$ such that the probability that $A(x, \mathrm{Leak}(x, y), i) = y_i$ is at most $1/3$.*

This definition allows for separate running times for the leakage-providing algorithm Leak and A, and in particular allows A to run in time that is sublinear in the length of a solution $y \in R(x)$.

▶ **Theorem 7.** *The following are equivalent:*

1. $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ *for all* $\epsilon > 0$.
2. $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$.
3. *For all $\epsilon > 0$ there exists a relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is $\mathsf{poly}(n)$-local $(\infty, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*
4. *For all $\epsilon > 0$ there exists a relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is $\mathsf{poly}(n)$-local $(2^{n^\epsilon}, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*
5. *For all $\epsilon > 0$ and $c \geq 1$ there exists a length-preserving relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is n^c-local $(n^c, \ell(n))$-leakage resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, where $n^{\Omega(1)} \leq \ell(n) \leq n - \omega(1)$ is polynomial-time computable.*

We only know Theorem 7 for the low-end of the derandomization spectrum due to the use of [2].

Techniques. We combine the approach of Liu and Pass in the leakage-resilient setting for BPP (see Section 2 for an overview) with constructions that are suitable for the AM setting.

In the hardness-to-derandomization direction, given a hard relation R, we use the value of $y \in R(x)$ as a basis for instantiating the nondeterministic version of the Shaltiel-Umans (SU) generator for prAM [18]. As the SU generator is originally meant for the the non-uniform setting, the original reconstruction argument only guarantees the existence of a small nondeterministic circuit that computes y in case the generator fails. Thus, in order to use the construction we need to "uniformize" the reconstruction procedure and show that such a circuit can be constructed by an efficient algorithm with non-adaptive oracle access to SAT. To obtain the results under hardness against learn-and-evaluate protocols, we employ the same approach but instead use a recursive version of the Miltersen-Vinodchandran generator [16], which we refer to as RMV and is due to Shaltiel and Umans [20]. RMV does not scale as well as SU but still suffices for obtaining Theorem 4. To employ the construction, we need to cast the RMV reconstruction process as a learn-and-evaluate protocol.

For the derandomization-to-hardness direction, similar to [15], we frame the problem of computing a leakage-resilient hard function as a $\mathsf{BPP}_{||}^{\mathsf{SAT}}$ search problem, and then make use of a search-to-decision reduction à la [8] together with the derandomization assumption. We first show that for a fixed x, a random choice r of $y \in R(x)$ suffices: With high probability, the first few algorithms Leak and A according to some canonical enumeration fail to compute r in the sense that $A(x, \mathrm{Leak}(x, r)) \neq r$. Then we show that checking whether a candidate r for $R(x)$ is hard (again w.r.t. the first few algorithms) can be done by a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ algorithm. To conclude the argument for $R \in \Sigma_2\mathsf{P}$, we guess-and-verify a "good" value of $y \in R(x)$ by exploiting the connection between $\mathsf{BPP}_{||}^{\mathsf{SAT}}$ and prAM [4] together with the derandomization assumption on prAM.

We remark that both directions of the Liu-Pass result for prBPP relativize, which immediately implies an equivalence between the derandomization $\mathsf{prBPP}^{\mathsf{SAT}} \subseteq \mathsf{P}^{\mathsf{SAT}}$ and the existence of leakage-resilient hard functions computable in $\mathsf{P}^{\mathsf{SAT}}$ that are hard on almost-all inputs against probabilistic algorithms with SAT oracle. Since our objective is to obtain equivalences with respect to derandomizing prAM, and it is unknown whether (mild) derandomization of prAM implies derandomization of $\mathsf{prBPP}^{\mathsf{SAT}}$, the relativization approach is insufficient for our purposes.

Organization. In Section 2, we develop the ideas behind our results and relate them to existing techniques. We start the formal treatment in Section 3 with definitions, notation, and other preliminaries. In Section 4, we develop the equivalence between leakage-resilient hardness on almost-all inputs and mild derandomization of prAM (Theorem 3), and the equivalence between mild derandomization of prAM and the existence of targeted hitting-set generators (Theorem 5). The development of the equivalence between leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ and against learn-and-evaluate protocols (Theorem 4) and the discussion of local leakage resilience (Theorem 7) are relegated to the appendix. The details of the uniformization of SU and proofs of secondary results are deferred to the full version.

2 Technical overview

In this section, we present an overview of the techniques underlying our results. In order to describe the Liu-Pass approach and ours, we start with an overview of the learning reconstructive paradigm used in prior hardness vs. randomness tradeoffs for BPP and AM.

We then provide an overview of the hardness-to-derandomization direction of the Liu-Pass result in a way that facilitates the transition to our results on the mild derandomization setting for AM.

Learning reconstructive paradigm. Pseudorandom generator constructions G typically have a hard function h as a basis (where how "hard" h is depends on the context) and produce a pseudorandom distribution G^h that depends on h. The proof of correctness for such generators is *reconstructive*: It presents an algorithm (the reconstructor) that, given access to any process D that distinguishes the distribution G^h from a truly random distribution, as well as to some additional information a, computes the hard function h. It is common for the additional information a to be composed of evaluations of h at a small number of points, in which case we also say that the reconstruction is *learning*. Thus, unless G^h "fools" an efficient randomized process P on input x, the function h can be reconstructed efficiently from the distinguisher $D(r) \doteq P(x, r)$ and the evaluations.

Targeted setting. One way to obtain a targeted PRG is to use a hardness-based (oblivious) pseudorandom generator construction and instantiate it with a function h_x that depends on the input x (which we view as a circuit capturing the behavior of the algorithm A on the actual input as a function of the random bit string) [5, 15, 23]. In the leakage-resilient setting and given a hard function f, we take h_x to be the function whose truth-table is $f(x)$. The leakage-providing algorithm Leak uses access to $f(x)$ to provide the answers to the learning queries, which are then fed into the reconstructor to compute h_x and thus $f(x)$.

Liu and Pass [15] employ the Nisan-Wigderson (NW) pseudo-random generator construction [17] combined with the locally-list-decodable encoding of [21], and instantiate the PRG with the function $h_x : \{0,1\}^{\log n} \to \{0,1\}$ that computes the mapping $i \mapsto f(x)_i$. In the reconstruction, the leakage-providing algorithm Leak uses access to h_x to answer the learning queries for the NW reconstruction algorithm as well to identify a "good" random string to be used in the list-decoding step and the position of h_x in the resulting list. This allows A to compute the value of $f(x)$ by running the NW reconstruction followed by the list-decoding process with the "good" random string, outputting the element in the correct position of the resulting list.

In porting the Liu-Pass approach to the mild derandomization setting for prAM, we follow a similar idea but replace the NW construction by other hardness-based pseudorandom generator (PRG) or hitting-set generator (HSG) constructions that exhibit the learning property: In the case of failure of the PRG/HSG on an input x, the underlying hard function h can be reconstructed efficiently from the values of h at a small number of points. In particular, we make use of the following HSGs that have the learning property and are tailored for Arthur-Merlin protocols:

- The construction by Shaltiel and Umans [20], which we dub SU and use to derive our main results under leakage-resilient hardness against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. This construction scales throughout the entire derandomization spectrum and, in particular, allows for obtaining tighter connections with blackbox derandomization via our local-leakage-resilient hardness results. To employ this generator, we present a uniform version for its reconstruction.
- A recursive version of the construction by Miltersen and Vinodchandran [16] due to Shaltiel and Umans [20], which we dub RMV and use to derive our results under hardness against a specific type of Arthur-Merlin protocol that we call a *learn-and-evaluate* protocol. The original construction, denoted MV, is geared toward the high end of the derandomization spectrum, whereas the recursive variant works in a broader range.

We remark that the SU construction scales better than RMV toward the low end, and thus we choose to employ SU for our main results. The disadvantage is that SU requires hardness against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ instead of against Arthur-Merlin protocols. However, as evidenced in Theorem 3, such a hardness assumption is also required in the mild setting.

We now describe the two constructions in more detail and how we employ them to obtain our results.

Shaltiel-Umans construction. The SU generator takes any low-degree polynomial $\hat{g} : \mathbb{F}^r \to \mathbb{F}$, which is usually obtained via a low-degree extension/Reed-Müller encoding (in our case, of $y \in R(x)$ of length n), and produces a set of strings of length m, where m is a parameter. The reconstruction for the generator shows the existence of a small (of size $\mathsf{poly}(m, \log n)$) nondeterministic circuit computing $\hat{g}$ in case the construction fails as a hitting-set generator.

The original reconstruction for the generator requires access to a few pieces of information in relation to the field $\mathbb{F}^r$ and the polynomial $\hat{g}$: A generating matrix M for the set $\mathbb{F}^r \setminus \{\vec{0}\}$, two "good" low-degree curves C_1 and C_2 over $\mathbb{F}^r$ and evaluations of $\hat{g}$ over points that depend on M, C_1 and C_2. For our purposes, the leakage-providing algorithm can compute the matrix M efficiently enough via a brute-force search. As for the curves C_1 and C_2, Shaltiel and Umans show that a random choice has the required properties with high probability, which allows Leak to sample those at random. Finally, the evaluations of $\hat{g}$ can be computed efficiently given access to the value of $y \in R(x)$ by having Leak compute the low-degree extension $\hat{g}$ of y.

In the nondeterministic setting, there is an extra complication: The reconstruction also requires oracle access to a list of r predictors $P_1, \ldots, P_r$, all of which can be obtained from a distinguisher D for the generator. As the predictors themselves are nondeterministic, it is unclear how, for example, negative predictions can be computed nondeterministically. Shaltiel and Umans use a strategy (originating in [6]) to approximately and nondeterministically evaluate the predictors when given as non-uniform additional input approximations up to $O(\log n)$ bits of precision for the probability that each predictor outputs 1 over a random choice of inputs for the generator. The original reconstruction argument can handle a small amount of error in these probabilities due to the use of error correction. This is where the SAT oracle comes in: We can efficiently obtain the approximations required to run the reconstruction in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.

In the end, we obtain a probabilistic version of the reconstruction with non-adaptive oracle access to SAT that, given oracle access to $\hat{g}$ and to a distinguisher D, outputs with high probability a small nondeterministic circuit C that computes $\hat{g}$. We then take Leak to be the algorithm that computes C using the low-degree extension of $y \in R(x)$ to answer the learning queries. Recall that C has size $\mathsf{poly}(m, \log n)$ and y has length n. By picking a small value of m, the amount of leakage is indeed small. We take A to be a circuit-evaluation algorithm that evaluates C to recover the value of y. Note that since C computes $\hat{g}$, in particular it computes the mapping $i \mapsto y_i$ very efficiently, and thus the reconstruction can also be cast into the local leakage-resilience framework.

Recursive Miltersen-Vinodchandran construction. The RMV generator also takes any low-degree polynomial $\hat{g} : \mathbb{F}^r \to \mathbb{F}$ and a parameter m and outputs a set of strings of length m.

The RMV reconstructor for $\hat{g}$ forms a *commit-and-evaluate protocol*, a notion introduced in [20] for this reason. It is an Arthur-Merlin protocol that consists of two phases. The first phase is the commitment phase, where Arthur and Merlin interact to produce a commitment π. The commitment is then given as input to the evaluation phase, in which Arthur and Merlin

compute evaluations of a function g_π that is determined by π and supposedly equals $\hat{g}$. As Merlin could cheat in the commitment phase, a key property of the protocol is its *resilience*: Given a commitment π, the evaluation protocol can only produce evaluations of a fixed function g_π, except with low probability. As g_π might differ from $\hat{g}$, one usually combines the construction with a checker (as in [10]) or a PCP system (as in [23]) to verify that $g_\pi = \hat{g}$.

In the commitment phase, Arthur samples a small (of size $\mathsf{poly}(m, \log n)$) set of points in $\mathbb{F}^r$ and Merlin provides evaluations of $\hat{g}$ at those points as the commitment π. With high probability over the set chosen by Arthur, given the correct commitment π as additional input and access to a distinguisher D, the evaluation protocol computes $\hat{g}$ with high confidence, no matter what strategy Merlin uses.

By replacing the interaction in the commitment phase by a probabilistic algorithm with oracle access to $\hat{g}$, the RMV reconstruction can be cast as a *learn-and-evaluate protocol*. A learn-and-evaluate protocol is a reconstructor for a function h that consists of two phases, where only the second phase requires access to the distinguisher D. The two phases are:

1. A learning phase, which is a randomized algorithm that can make queries to h and outputs a string, which we refer to as a sketch.
2. An evaluation phase, which is a promise Arthur-Merlin protocol with access to D that takes a sketch and an evaluation point z as input, and is supposed to output $h(z)$.

With high probability, given the correct answers to the queries, the learning phase should output a sketch such that the evaluator with access to a distinguisher D is complete and sound.

The learn-and-evaluate version of the reconstructor naturally fits the leakage-resilient hardness-to-derandomization framework, where the leakage-providing algorithm Leak is a probabilistic algorithm with access to $y \in R(x)$ that samples the set of points and produces evaluations of the low-degree extension $\hat{g}$ of y on those points and A is an Arthur-Merlin protocol that takes those evaluations as additional input to compute y. The amount of leakage is upper bounded by $\mathsf{poly}(m, \log n)$, which can be made small in relation to $|y| = n$ by picking a sufficiently small m.

3 Preliminaries

In this section, we present preliminary definitions and results that are necessary for developing our contributions. We start by defining targeted hitting-set generators, then move on to low-degree extensions and finally present some results on the class $\mathsf{BPP}_{||}^{\mathsf{SAT}}$. We assume familiarity with standard notions such as circuits and complexity classes such as NP, AM, and prAM. We measure circuit size by the number of gates. Most of our results extend to "robust" time bounds: We define a set of time bounds $\mathcal{T}$ as robust if for all polynomials p, q it holds that if $T \in \mathcal{T}$, then $p(n) \cdot T(q(n)) \in \mathcal{T}$.

Targeted hitting-set generators. Without loss of generality, Arthur-Merlin protocols have perfect completeness [7]. In this case, targeted hitting-set generators, the one-sided error variant of targeted PRGs, suffice for derandomization. We may naturally view an Arthur-Merlin protocol as a nondeterministic computation that receives an input x and a random sequence ρ, guesses a response from Merlin and accepts if and only if the final deterministic verification of Arthur accepts. Since such a computation can only make a mistake on a negative instance x, we wish for a generator that hits rejection, i.e., it produces a set S such that there is $\rho \in S$ that is rejected by the nondeterministic computation. As hitting-set generators are usually considered for the accepting side of computations, we instead consider generators for co-nondeterministic computations.

A co-nondeterministic computation on the random bits ρ for a given input x can be captured by a co-nondeterministic circuit. For this reason, in the context of derandomizing Arthur-Merlin protocols, we let targeted hitting-set generators take as input a co-nondeterministic circuit D that accepts at least a fraction $1/2$ of its inputs, and output a set of strings that "hits" D. If the co-nondeterministic computation runs in time $t(n)$ for inputs x of length n, then the co-nondeterministic circuit has size at most $t(n)^2$ for all but finitely many n. To accommodate different computational models for *generating* the hitting sets (including nondeterministic ones), we define targeted generators based on a relation between a circuit D and a hitting-set S. In the following definition, we let $\mathcal{C}$ denote a machine model and $\mathcal{C}\mathsf{TIME}[T]$ denote computational problems computable by machines in $\mathcal{C}$ that run in time T.

▶ **Definition 8.** *Let H be an algorithm that computes a relation $R \in \mathcal{C}\mathsf{TIME}[T(m)]$ between co-nondeterministic circuits of size m and sets of strings of length m. We say that H is a targeted hitting-set generator for co-nondeterministic circuits computable in $\mathcal{C}\mathsf{TIME}[T(m)]$ if the following two conditions hold for all sufficiently large $m \in \mathbb{N}$:*

- *For all co-nondeterministic circuits D of size m, there exists a non-empty S such that $(D, S) \in R$.*
- *For all co-nondeterministic circuits D of size m that accept at least a $1/2$ fraction of their inputs, it holds that for every $S \in R(D)$ there exists $\rho \in S$ such that $D(\rho)$ accepts.*

For some classes of algorithms, such as nondeterministic algorithms, the notion of computing a relation is intuitive: Upon receiving an input, the algorithm eventually halts and produces an output in the relation on every accepting computation path. For other classes of algorithms, such as $\Sigma_2\mathsf{P}$, the notion of computing a relation is perhaps less intuitive. We say that a Σ_2-algorithm N computes a relation R if for all $x \in \{0,1\}^*$, any accepting configuration of $N(x)$ outputs a value y such that $(x, y) \in R$.

We remark that the existence of a targeted hitting-set generator for co-nondeterministic circuits suffices for derandomizing all of prAM. For future reference, we include a statement and proof in the Σ_2 setting.

▶ **Proposition 9.** *Assume that there exists a targeted hitting-set generator H for co-nondeterministic circuits computable in $\Sigma_2\mathsf{TIME}[T(m)]$. Then $\mathsf{prAM} \subseteq \bigcup_{k\in\mathbb{N}} \Sigma_2\mathsf{TIME}[T(n^k)]$.*

Proof. Let $\Pi \in \mathsf{prAM}$, and let P be an Arthur-Merlin protocol (with perfect completeness) that runs in time n^k for some constant k and decides Π. It is standard that we can obtain from P and a sufficiently large input $x \in \{0,1\}^*$, in time $O(|x|^{2k})$, a co-nondeterministic circuit $D_{P,x}$ of size at most $|x|^{2k}$ such that:

$$x \in \Pi_Y \implies \Pr_r[D_{P,x}(r) = 1] = 0.$$

$$x \in \Pi_N \implies \Pr_r[D_{P,x}(r) = 1] \geq 1/2.$$

The Σ_2-simulation for Π works as follows on input x: First, it computes $D_{P,x}$, and guesses a set S output by $H(D_{P,x})$. Using another nondeterministic guess, the simulation verifies that $D_{P,x}$ rejects every $\rho \in S$, rejecting otherwise. In parallel, the simulation verifies, using an existential and a universal guess, that the guessed set S is an output of H for $D_{P,x}$, rejecting otherwise.

If $x \in \Pi_Y$, then $D_{P,x}$ rejects every $\rho \in S$ for any S output by $H(D_{P,x})$, in which case the overall simulation accepts. If $x \in \Pi_N$, then correctness of H implies that $D_{P,x}$ accepts some $\rho \in S$ for every S output by $H(D_{P,x})$, in which case the overall simulation rejects. The running time for the simulation is $T(n^{2k})$, which completes the proof. ◀

Low-degree extension. We also need standard low-degree extensions. Let $g : \{0,1\}^\ell \to \{0,1\}$ be a function, $\mathbb{F} = \mathbb{F}_p$ be the field with p elements (for prime p), h and r integers such that $h^r \geq 2^\ell$, and $[h]^r$ the r-tuples of elements in $\{1,\ldots,h\}$. The low-degree extension of g with respect to p, h, r is the unique r-variate polynomial $\hat{g} : \mathbb{F}^r \to \mathbb{F}$ with degree $h-1$ in each variable, for which $\hat{g}(\vec{v}) = g(y)$ for all $\vec{v} \in [h]^r$ representing a $y \in \{0,1\}^\ell$, and $\hat{g}(\vec{v}) = 0$ for $\vec{v} \in [h]^r$ that do not represent a string y. The total degree of $\hat{g}$ is $\Delta = hr$ and $\hat{g}$ is computable in time $\mathsf{poly}(h^r, \log p, r)$ given oracle access to g.

Non-adaptive oracle access to SAT. We first define the class $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$: We say that a promise problem $\Pi = (\Pi_Y, \Pi_N)$ is in $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$ if there exists a probabilistic algorithm M with non-adaptive oracle access to SAT such that for all $x \in \{0,1\}^*$:

$$x \in \Pi_Y \implies \Pr[M(x) = 1] \geq 2/3.$$
$$x \in \Pi_N \implies \Pr[M(x) = 1] \leq 1/3.$$

By non-adaptive oracle access, we mean that the queries made by M cannot depend on the answers to previous queries, i.e., the queries must all be made in parallel.

Arthur-Merlin protocols can be simulated by probabilistic algorithms with oracle access to SAT: To simulate the interaction, we sample a random sequence for Arthur and query the SAT oracle on whether there exists a response of Merlin that would lead to acceptance. In a similar fashion, $\mathsf{P}^{\mathsf{prAM}}_{||}$ can be simulated in $\mathsf{BPP}^{\mathsf{SAT}}_{||}$, the class of problems decidable by bounded-error probabilistic polynomial-time algorithms with non-adaptive oracle access to SAT. In fact, a converse also holds and is crucial in allowing our results to work under the assumption that prAM can be mildly derandomized.

▶ **Lemma 10** ([4]). $\mathsf{prBPP}^{\mathsf{SAT}}_{||} \subseteq \mathsf{P}^{\mathsf{prAM}}_{||}$.

In Lemma 10, the deterministic machines with oracle access to prAM on the right-hand side are guaranteed to work correctly irrespective of how the queries outside of the promise are answered, even if those queries are answered inconsistently, i.e., different answers may be given when the same query is made multiple times.

4 Equivalences for mild derandomization of prAM

In this section, we develop our main results, equivalences between mild derandomization of prAM and the existence of leakage-resilient hard functions/relations. We first state a more general version of Theorem 3 for classes $\mathcal{C}$ such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. For such classes, we obtain an equivalence with respect to a hard function.

▶ **Theorem 11.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prAM} \subseteq \mathcal{C}$.
- *There exists a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$.*

Theorem 11 follows from Theorem 14 in Section 4.1 and Theorem 19 in Section 4.2. Examples of classes for which Theorem 11 applies are P^{NP} and $\mathsf{ZPP}^{\mathsf{NP}}$ and their time-$2^{\mathsf{polylog}(n)}$ and time-$2^{n^{o(1)}}$ variants.

In the case of $\Sigma_2\mathsf{P}$, we state a more general equivalence with respect to a hard relation.

▶ **Theorem 12.** *Let $\mathcal{T}$ be a robust set of time bounds. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prAM} \subseteq \bigcup_{T \in \mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$.
- *There exists $T' \in \mathcal{T}$ and a total length-preserving relation $R \in \Sigma_2\mathsf{TIME}[T'(n)]$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

Theorem 12 follows from Theorem 15 in Section 4.1 and Theorem 21 in Section 4.2. Similar to Theorem 11, Theorem 12 applies to the time-$\mathsf{poly}(n)$, time-$2^{\mathsf{polylog}(n)}$ and time-$2^{n^{o(1)}}$ variants of $\Sigma_2\mathsf{P}$. Theorem 3 follows by instantiating Theorems 11 and 12 with polynomial time bounds.

4.1 From leakage-resilient hardness to derandomization

Assuming the existence of a leakage-resilient hard function/relation against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, we show that we can obtain mild derandomization of prAM. Our approach is to instantiate the Shaltiel-Umans generator with the value of $f(x)$ (or some $y \in R(x)$ in case of a hard relation R). In case the generator fails, there exists a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ algorithm (the reconstructor) that given $f(x)$ as input outputs a small nondeterministic circuit computing the mapping $i \mapsto f(x)_i$. We take the reconstructor as the leakage-providing algorithm Leak and define A as a circuit-evaluation algorithm that evaluates the circuit output by Leak.

The next lemma presents a uniform version of the SU reconstruction that is particularly suited to our objectives. See the full version for a proof.

▶ **Lemma 13.** *There exists a deterministic algorithm H_{det} and a probabilistic algorithm* Learn *with non-adaptive oracle access to* SAT *such that at least one of the following holds for every $y \in \{0,1\}^n$, $m \in \mathbb{N}$ and co-nondeterministic circuit D of size m that accepts at least half of its inputs:*

1. *$H_{det}(1^m, y)$ outputs a set that hits D.*
2. *With probability at least $2/3$,* $\mathrm{Learn}(1^m, y, D)$ *outputs a nondeterministic circuit C that computes the mapping $i \mapsto y_i$.*

H_{det} and Learn *run in time* $\mathsf{poly}(m, n)$ *and C has size* $\mathsf{poly}(m, \log n)$*. Moreover,* Learn *and C only need blackbox access to the deterministic predicate that underlies D.*

We are now ready to prove the hardness-to-derandomization direction of Theorem 11.

▶ **Theorem 14.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$ and $\mathsf{NP} \subseteq \mathcal{C}$. There exists a constant $c \geq 1$ such that the following holds. Assume that there exist a constant $0 < \epsilon < 1$ and a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. Then there exists a targeted hitting-set generator H for co-nondeterministic circuits computable in $\mathcal{C}$, implying that $\mathsf{prAM} \subseteq \mathcal{C}$.*

Proof. Fix a binary string representation for co-nondeterministic circuits that describes a circuit of size m by a string of length $m' = \Theta(m \log m)$. The generator H, on input a string x of length m' describing a co-nondeterministic circuit D_x of size m, sets $n = m^a$ for a sufficiently large constant a to be defined later and sets $x' = x0^{n-m'}$. It then computes $f(x')$ and instantiates the generator H_{det} of Lemma 13, outputting the set $S_x = H_{\mathrm{det}}(1^m, f(x'))$. Computing $f(x')$ can be done in $\mathcal{C}$ by the assumption that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$ and since $|x'| = \mathsf{poly}(m)$, and computing S_x from $f(x')$ takes deterministic time $\mathsf{poly}(m, n) = \mathsf{poly}(m)$, and therefore the generator H is computable in $\mathcal{C}$.

Assume, with the intent of deriving a contradiction, that H fails as a targeted HSG for co-nondeterministic circuits. This means that there exists an infinite set $\mathcal{D}$ of co-nondeterministic circuits that accept at least a $1/2$ fraction of their inputs such that $H(D)$ fails to hit every $D \in \mathcal{D}$. We show that there exist probabilistic algorithms Leak and A with non-adaptive oracle access to SAT running in time n^c for a constant c to be defined later such that for infinitely many x', $\mathrm{Leak}(x', f(x'))$ produces $|x'|^\epsilon$ bits of leakage and $A(x', \mathrm{Leak}(x', f(x')))$ computes $f(x')$ with high probability.

The algorithm Leak parses the input $x' \in \{0,1\}^n$ as $x' = x0^{n-m'}$ for $m' = \Theta(m \log m)$ and $m = n^{1/a}$. If x' is not of the expected type, it outputs 0^n. Then, it outputs $\mathrm{Learn}(1^m, f(x'), D_x)$, where D_x is the circuit described by x. The algorithm A is a circuit evaluation algorithm that, on input x' of length n and a nondeterministic circuit C, makes parallel queries to its SAT oracle about the output of C for each $i \in [n]$, outputting the concatenation of those (to hopefully obtain $f(x')$).

Notice that we measure the running times of Leak and A in terms of $n = m^a$. By Lemma 13, Leak runs in time $\mathsf{poly}(m, n) = \mathsf{poly}(n) \leq n^c$ for a sufficiently large constant c. By the same lemma, n^c serves as an upper bound for the running time of A, which is $n \cdot \mathsf{poly}(m, \log n)$. Moreover, the amount of leakage is $\mathsf{poly}(m, \log n) \leq (m \log n)^k$ for a sufficiently large constant k. By taking $a = 2k/\epsilon$, this is at most n^ϵ.

As for correctness, Lemma 13 guarantees that for all x such that $H(D_x)$ fails to hit D_x, $\mathrm{Learn}(1^m, x', f(x'))$ for $x' = x0^{n-m'}$ outputs with probability at least $2/3$ a circuit C that computes the mapping $i \mapsto f(x')_i$. In that case, A with inputs x' and C outputs $f(x')$. The conclusion $\mathsf{prAM} \subseteq \mathcal{C}$ follows as in Proposition 9: Given an Arthur-Merlin protocol P witnessing $\Pi \in \mathsf{prAM}$ and an input x, we construct $D_{P,x}$ and compute a hitting set $S = H(D_{P,x})$. Finally, we use the fact that $\mathsf{NP} \subseteq \mathcal{C}$ to verify in $\mathcal{C}$ that $D_{P,x}$ rejects all $\rho \in S$, rejecting otherwise. ◀

Essentially the same argument establishes the result for a leakage-resilient hard relation $R \in \Sigma_2\mathsf{TIME}[T(n)]$.

▶ **Theorem 15.** *There exists a constant $c \geq 1$ such that the following holds. Assume that there exist a constant $0 < \epsilon < 1$ and a length-preserving relation $R \in \Sigma_2\mathsf{TIME}[T(n)]$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. Then there exists targeted hitting-set generator H for co-nondeterministic circuits computable in $\Sigma_2\mathsf{TIME}[T(\mathsf{poly}(m)) + \mathsf{poly}(m)]$, implying that $\mathsf{prAM} \subseteq \bigcup_{k \in \mathbb{N}} \Sigma_2\mathsf{TIME}[T(n^k)]$.*

Proof (sketch). The proof follows the argument of Theorem 14 closely. Instead of computing $H_{\mathrm{det}}(1^m, f(x'))$, the generator H guesses and verifies $y \in R(x')$ and then outputs $H_{\mathrm{det}}(1^m, y)$, which leads to a generator computable in time $O(T(m^a) + \mathsf{poly}(m))$. The reconstruction is identical, and allows for computing $y \in R(x')$ given a small amount of leakage on y. ◀

We remark that the arguments for Theorems 14 and 15 show, in particular, that it is possible to compute $f(x')$ (or $y \in R(x')$) *locally* in time $\mathsf{poly}(m, \log n)$ by letting A be a regular circuit evaluation algorithm (instead of one that outputs the concatenation of evaluating the input circuit on every i).

4.2 From derandomization to leakage-resilient hardness

We first establish the derandomization-to-hardness direction of Theorem 11. As mentioned in Section 1, we frame the problem of computing a leakage-resilient hard function as a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem and then make use of a search-to-decision reduction as in [8].

We start by defining a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem.

▶ **Definition 16.** *Let R_Y and R_N be two disjoint binary relations. We say that (R_Y, R_N) is a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem if the following two conditions hold.*

1. *The decisional promise problem represented by (R_Y, R_N) is in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$; that is, there exists a probabilistic polynomial-time algorithm V with non-adaptive oracle access to SAT such that for every $(x, y) \in R_Y$ it holds that $\Pr[V(x, y) = 1] \geq 2/3$ and for every $(x, y) \in R_N$ it holds that $\Pr[V(x, y) = 1] \leq 1/3$.*
2. *There exists a probabilistic polynomial-time algorithm A with non-adaptive oracle access to SAT such that, for every x for which $R_Y(x) \neq \emptyset$, it holds that $\Pr[A(x) \in R_Y(x)] \geq 2/3$, where $R_Y(x) = \{y \mid (x, y) \in R_Y\}$.*

We observe that the search-to-decision strategy developed in [8] ports over to $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ problems.

▶ **Proposition 17.** *For every $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem (R_Y, R_N), there exists a binary relation R such that $R_Y \subseteq R \subseteq (\{0,1\}^* \times \{0,1\}^*) \setminus R_N$ and solving the search problem of R is (adaptively) deterministically polynomial-time reducible to some decisional problem in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

The following standard result will be useful to compose computations that have non-adaptive access to a SAT oracle.

▶ **Proposition 18** (See e.g., [19, Lemma 7.2])**.** *There exists a non-adaptive SAT-oracle algorithm M with the following behavior. On input $x \in \{0,1\}^n$ and the description of two non-adaptive SAT-oracle algorithms M_1 and M_2 such that M_1 runs in time time $t_1(n)$ and produces outputs of length $t_1(n)$ and M_2 takes inputs of length $t_1(n)$ and runs time $t_2(n)$, M runs in time $\mathsf{poly}(n, t_1(n), t_2(n))$ and outputs $M_2(M_1(x))$.*

Now, we are ready to prove Theorem 19.

▶ **Theorem 19.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. If $\mathsf{prAM} \subseteq \mathcal{C}$, then for all constants c and $0 < \epsilon < 1$ there exists a length-preserving function $f \in \mathcal{C}$ that is $(n^c, n - \omega(1))$-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, where $\omega(1)$ is polynomial-time computable.*

Proof. Our approach is to cast computing a leakage-resilient hard function f as a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem, which allows us to instantiate Proposition 17 and show that there is a $\mathsf{P}^{\mathsf{prBPP}_{||}^{\mathsf{SAT}}} \subseteq \mathsf{P}^{\mathsf{P}_{||}^{\mathsf{prAM}}} \subseteq \mathsf{P}^{\mathsf{prAM}}$ algorithm that solves it. Finally, we use the derandomization assumption together with the assumption on $\mathcal{C}$ to conclude that f is computable in $\mathcal{C}$.

First, we argue that for any x, a random choice of $f(x)$ is hard w.r.t. a fixed pair (Leak, A) of probabilistic algorithms with non-adaptive oracle access to SAT. Fix a constant c, an input $x \in \{0,1\}^n$ and a polynomial-time computable function $\nu(n) = \omega(1)$. Let ρ_{Leak} and ρ_A denote the random bits input to Leak and A, respectively. Let

$$P(r, \rho_{\mathrm{Leak}}, \rho_A) \equiv |\mathrm{Leak}(x, r; \rho_{\mathrm{Leak}})| \leq \ell \wedge A(x, \mathrm{Leak}(x, r; \rho_{\mathrm{Leak}}); \rho_A) = r,$$

for $\ell = n - \nu(n)$. We say that r fails if $\Pr_{\rho_{\mathrm{Leak}}, \rho_A}[P(r, \rho_{\mathrm{Leak}}, \rho_A)] \geq 1/6$.

We have that

$$\begin{aligned}
\Pr_r[r \text{ fails}] &= \Pr_r[\Pr_{\rho_{\text{Leak}},\rho_A}[P(r,\rho_{\text{Leak}},\rho_A)] \geq 1/6] && \text{[definition of failing } r\text{]} \\
&\leq 6 \cdot \mathbb{E}_r[\Pr_{\rho_{\text{Leak}},\rho_A}[P(r,\rho_{\text{Leak}},\rho_A)] && \text{[Markov's inequality]} \\
&= 6 \cdot \mathbb{E}_{r,\rho_{\text{Leak}},\rho_A}[P(r,\rho_{\text{Leak}},\rho_A)] && \text{[expectation of indicator variable]} \\
&= 6 \cdot \mathbb{E}_{\rho_{\text{Leak}},\rho_A}[\mathbb{E}_r[P(r,\rho_{\text{Leak}},\rho_A)]] && \text{[reordering random bits]} \\
&< 6\frac{2^{\ell+1}}{2^n} && \text{[pigeonhole argument]}
\end{aligned}$$

The pigeonhole argument is that, after fixing the random bits ρ_{Leak} and ρ_A, for each of the at most $2^{\ell+1}$ strings y of length at most ℓ, among all the strings r that Leak maps to y, there is at most one that A maps back to r. Setting $\ell = n - \nu(n)$ implies that the probability that a string r is "bad" is at most $12/2^{\nu(n)}$.

We then define the search problem (R_Y, R_N) such that $(x,r) \in R_Y$ if $|x| = |r| = n$ and for the first $\nu(n)$ pairs of probabilistic machines with non-adaptive oracle access to SAT (Leak, A) clocked to run in time n^c, it holds that

$$\Pr_{\rho_{\text{Leak}},\rho_A}[|\text{Leak}(x,r;\rho_{\text{Leak}})| \leq \ell \wedge A(x,\text{Leak}(x,r);\rho_A) = r] < \frac{1}{6}. \tag{1}$$

As for "no" instances, $(x,r) \in R_N$ if for at least one pair out of the first $\nu(n)$ (Leak, A), equation (1) with $1/6$ replaced by $1/3$ does not hold.

Now, we show that this problem is a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem. On input $x \in \{0,1\}^n$, the algorithm for finding a solution samples a random $r \in \{0,1\}^n$. By a union bound over the $\nu(n)$ many algorithms Leak and A and the fact that r fails for a particular pair with probability at most $12/2^{\nu(n)}$, it holds that the solution-finding algorithm succeeds with probability at least $2/3$ for sufficiently large n. On input (x,r), the verification algorithm enumerates the first $\nu(n)$ probabilistic machines with non-adaptive oracle access to SAT, Leak and A, all clocked to run in time n^c. It then estimates the value

$$p_{\text{Leak},A} = \Pr_{\rho_{\text{Leak}},\rho_A}[|\text{Leak}(x,r;\rho_{\text{Leak}})| \leq \ell \wedge A(x,\text{Leak}(x,r);\rho_A) = r]$$

up to error $1/12$ and with failure probability at most $1/n$ for each pair (Leak, A). By a standard Chernoff bound, it suffices to perform the following steps a polynomial (in n) number of times per pair: Let Leak' be the algorithm that, on input x and a random sequence ρ_{Leak} for Leak, computes an output $y = \text{Leak}(x,r;\rho_{\text{Leak}})$ and outputs (x,y). Let A' be an algorithm that, on input x, y and random sequence ρ_A, rejects if $|y| > \ell$ and outputs $A(x,y;\rho_A)$ otherwise, and let M be the algorithm of Proposition 18. Sample random strings ρ_{Leak} and ρ_A for Leak and A, respectively, compute $A'(x,\text{Leak}'(x,r;\rho_{\text{Leak}});\rho_A)$ by feeding M inputs $x, r, \rho_{\text{Leak}}, \rho_A$ and the codes of Leak' and A' and compare the output with r. Each such execution requires time $\mathsf{poly}(n)$ for a total running time of $\mathsf{poly}(n)$. Moreover, all oracle queries made by M can be made in parallel.

After estimating the average acceptance probability for each pair (Leak, A), the verification algorithm outputs 1 if the estimated values are less than $1/4$ (the midpoint between $1/6$ and $1/3$) for all pairs of algorithms. By a union bound over the $\nu(n)$ pairs of algorithms, the algorithm accepts $(x,r) \in R_Y$ and rejects $(x,r) \in R_N$ with probability at least $2/3$.

Finally, we use Proposition 17 together with the derandomization assumption and the assumption on $\mathcal{C}$ to conclude that there is a function $f \in \mathcal{C}$ that solves (R_Y, R_N), i.e., f is a leakage-resilient hard function. ◀

The proof of Theorem 19 shows that, given x, verifying whether a candidate r for $R(x)$ is hard w.r.t. the first $\nu(n)$ algorithms Leak, A can be done in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. The proof also shows that a "good" r always exists for sufficiently large n. To extend Theorem 19 to obtain a hard relation $R \in \Sigma_2$, we first show that a Σ_2-derandomization assumption on prAM implies the same derandomization for $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. See the full version for a proof.

▶ **Proposition 20.** *Let $\mathcal{T}$ be a robust set of time bounds. If $\mathsf{prAM} \subseteq \bigcup_{T \in \mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$, then $\mathsf{prBPP}_{||}^{\mathsf{SAT}} \subseteq \bigcup_{T \in \mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$.*

Under a Σ_2-derandomization assumption for prAM, Proposition 20 allows a Σ_2 algorithm to compute a leakage-resilient hard relation R by guessing a candidate y and verifying in Σ_2 that it is a "good" solution. We therefore establish Theorem 21.

▶ **Theorem 21.** *Let $\mathcal{T}$ be a robust set of time bounds. If $\mathsf{prAM} \subseteq \bigcup_{T \in \mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$, then for every constant c there exists a total length-preserving relation $R \in \Sigma_2\mathsf{TIME}[T']$ for some $T' \in \mathcal{T}$ that is $(n^c, n - \omega(1))$-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, where $\omega(1)$ is polynomial-time computable.*

4.3 Targeted hitting-set generators from derandomization

In this section, we show that mild derandomization of prAM implies the existence of targeted hitting-set generators that suffice to obtain the original derandomization result. The results follow as consequences of the equivalence between leakage-resilient hardness and derandomization since we show the hardness-to-derandomization direction by constructing such targeted generators.

▶ **Corollary 22.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. If $\mathsf{prAM} \subseteq \mathcal{C}$, then there exists a targeted hitting-set generator for co-nondeterministic circuits computable in $\mathcal{C}$.*

Proof. By Theorem 14, there is a constant c such that if there is a length-preserving function $f \in \mathcal{C}$ that is $(n^c, n^{1/2})$-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, then there is a targeted HSG as in the conclusion. By Theorem 19, the assumption $\mathsf{prAM} \subseteq \mathcal{C}$ implies the existence of f as required. ◀

Corollary 23 is established in the exact same way.

▶ **Corollary 23.** *Let $\mathcal{T}$ be a robust set of time bounds. If $\mathsf{prAM} \subseteq \bigcup_{T \in \mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$, then there exists a targeted hitting-set generator for co-nondeterministic circuits computable in $\Sigma_2\mathsf{TIME}[T']$ for $T' \in \mathcal{T}$.*

Theorem 5 follows by combining Corollaries 22 and 23 with polynomial time bounds.

References

1. Leonard Adleman. Two theorems on random polynomial time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978. doi:10.1109/SFCS.1978.37.
2. Barış Aydınlıoğlu and Dieter van Melkebeek. Nondeterministic circuit lower bounds from mildly derandomizing Arthur-Merlin games. *Computational Complexity*, 26(1):79–118, 2017. doi:10.1007/s00037-014-0095-y.
3. László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993. doi:10.1007/BF01275486.
4. Venkatesan T. Chakaravarthy and Sambuddha Roy. Arthur and Merlin as oracles. *Computational Complexity*, 20(3):505–558, 2011. doi:10.1007/s00037-011-0015-3.

5 Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *Symposium on Foundations of Computer Science (FOCS)*, pages 125–136, 2022. doi:10.1109/FOCS52979.2021.00021.

6 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993. doi:10.1137/0222061.

7 Martin Furer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stahis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989.

8 Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 191–232. Springer, 2011. Part of the Lecture Notes in Computer Science book series (LNCS, volume 6650). doi:10.1007/978-3-642-22670-0_20.

9 Oded Goldreich. Two comments on targeted canonical derandomizers. In *Computational Complexity and Property Testing: On the Interplay Between Randomness and Computation*, pages 24–35. Springer, 2020. doi:10.1007/978-3-030-43662-9_4.

10 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3):85–130, 2003. doi:10.1007/s00037-003-0178-7.

11 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002. doi:10.1016/S0022-0000(02)00024-7.

12 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing (STOC)*, page 220–229, 1997. doi:10.1145/258533.258590.

13 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

14 Yanyi Liu and Rafael Pass. Characterizing derandomization through hardness of Levin-Kolmogorov complexity. In *Computational Complexity Conference (CCC)*, volume 234, pages 35:1–35:17, 2022. doi:10.4230/LIPIcs.CCC.2022.35.

15 Yanyi Liu and Rafael Pass. Leakage-Resilient Hardness vs Randomness. In *Computational Complexity Conference (CCC)*, volume 264, pages 32:1–32:20, 2023. doi:10.4230/LIPIcs.CCC.2023.32.

16 Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur–Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005. doi:10.1007/s00037-005-0197-7.

17 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

18 Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005. doi:10.1145/1059513.1059516.

19 Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006. doi:10.1007/s00037-007-0218-9.

20 Ronen Shaltiel and Christopher Umans. Low-end uniform hardness versus randomness tradeoffs for AM. *SIAM Journal on Computing*, 39(3):1006–1037, 2009. doi:10.1137/070698348.

21 Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom Generators without the XOR Lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

22 Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003. doi:10.1016/S0022-0000(03)00046-1.

23 Dieter van Melkebeek and Nicollas Mocelin Sdroievski. Instance-wise hardness versus randomness tradeoffs for Arthur-Merlin protocols. In *Computational Complexity Conference (CCC)*, volume 264, pages 17:1–17:36, 2023. doi:10.4230/LIPIcs.CCC.2023.17.

24 R. Ryan Williams. Natural proofs versus derandomization. *SIAM Journal on Computing*, 45(2):497–529, 2016. doi:10.1137/130938219.

Appendices

In Appendix A, we develop the equivalence between leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ and against learn-and-evaluate protocols (Theorem 4). In Appendix B, we present our results relating local leakage resilience to non-uniform lower bounds (Theorem 7).

A Leakage-resilient hardness against learn-and-evaluate protocols

In this appendix, we prove that the hardness-to-derandomization direction of Theorems 11 and 12 holds under hardness against learn-and-evaluate protocols, a specific type of Arthur-Merlin protocol. By combining such a result with the derandomization-to-hardness direction of Theorems 11 and 12, we obtain an equivalence in the leakage-resilient hardness setting between hardness against learn-and-evaluate protocols and hardness against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.

We start with some definitions. A *learn-and-evaluate protocol* is composed of two phases: A *learning* phase followed by an *evaluation* phase. In the learning phase, a probabilistic algorithm makes queries to a function f and produces an output (which we call a sketch). The evaluation phase then consists of an Arthur-Merlin protocol that computes $f(x)$ correctly on every input x when given the sketch as additional input. To define this notion, we first define what it means for an Arthur-Merlin protocol to produce an output.

▶ **Definition 24** (Arthur-Merlin protocol with output). *Let P be an Arthur-Merlin protocol. We say that on a given input $x \in \{0,1\}^*$:*

- *P outputs v with completeness c if there exists a Merlin strategy such that the probability that P succeeds and outputs v is at least c. In symbols: $(\exists M)\Pr[P(x,M)=v] \geq c$.*
- *P outputs v with soundness s if, no matter what strategy Merlin uses, the probability that P succeeds and outputs a value other than v is at most s. In symbols: $(\forall M)\Pr[P(x,M) \notin \{v,\perp\}] \leq s$.*

For a given function $f : X \to \{0,1\}^*$ where $X \subseteq \{0,1\}^*$, we say that P computes f with completeness $c(n)$ and soundness $s(n)$ if on every input $x \in X$, P outputs $f(x)$ with completeness $c(|x|)$ and soundness $s(|x|)$.

Now, we define learn-and-evaluate protocols.

▶ **Definition 25** (Learn-and-evaluate protocol). *A learn-and-evaluate protocol P consists of a probabilistic algorithm A_{learn} and an Arthur-Merlin protocol P_{eval}. Let $g : X \to \{0,1\}^*$ where $X \subseteq \{0,1\}^*$. We say that P computes g with error $e(n)$ for completeness $c(n)$ and soundness $s(n)$ if on every input $x \in X$ of length n the following hold: The probability over the randomness of A_{learn} that P_{eval} with input x and additional input $\pi = A_{learn}^{f}(1^n)$ outputs $f(x)$ with completeness $c(n)$ and soundness $s(n)$ is at least $1 - e(n)$.*

We define (T,ℓ)-leakage-resilient hardness on almost-all inputs against learn-and-evaluate protocols in a completely analogous way to hardness against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ (Definition 1), where a probabilistic algorithm $\mathrm{Learn}(x,y)$ (which we take to be A_{learn}^{y}) takes on the role of Leak, running in time T and producing, on input x of length n, an output of length at most $\ell(n)$, and P_{eval} takes on the role of algorithm A, also running in time T.

We make use of the following lemma, which hinges on the RMV generator. See the full version for a proof.

▶ **Lemma 26.** *There exists a deterministic algorithm H_{det}, a probabilistic algorithm* Learn *and an Arthur-Merlin protocol P_{eval} such that at least one of the following holds for every $y \in \{0,1\}^n$, $m \in \mathbb{N}$ and co-nondeterministic circuit D of size m that accepts at least half of its inputs:*

1. *$H_{det}(1^m, y)$ outputs a set that hits D.*
2. *With probability at least $2/3$, $\mathrm{Learn}(1^m, y)$ outputs a sketch π such that $P_{eval}(\pi, D, \cdot)$ computes the mapping $i \mapsto y_i$ with completeness 1 and soundness $1/n^2$.*

H_{det} and A_{learn} run in time $\mathsf{poly}(n, m)$, and A_{learn} produces a sketch π of length $\mathsf{poly}(m, \log n)$. P_{eval} runs in time $(m \cdot \log n)^{O(\log^2 r)}$ for $r = O(\log n / \log m)$. Moreover, P_{eval} only needs blackbox access to the deterministic predicate that underlies D.

We remark that the RMV reconstruction does not scale as well as the SU reconstruction, where the analogue of P_{eval} runs in time $\mathsf{poly}(m, \log n)$.

We now state the hardness-to-derandomization result that we obtain from Lemma 26.

▶ **Theorem 27.** *Theorems 14 and 15 continue to hold when $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ is replaced by learn-and-evaluate protocols.*

Proof. The proof follows closely that of Theorems 14 and 15 but uses the generator H_{det} of Lemma 26 instead of Lemma 13. For the reconstructor's running time, since we pick $n = \mathsf{poly}(m)$, we can upper bound the running time of Leak (which equals the algorithm Learn of Lemma 26) and the leakage-receiving protocol (which invokes P_{eval} n times) by n^c for some constant c. The bound on the leakage is calculated in the exact same way. ◀

As a consequence, we obtain equivalence between hardness on almost-all inputs against polynomial-time learn-and-evaluate protocols and leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. For simplicity, we state the result for classes $\mathcal{C}$ such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$, but it holds for Σ_2 as well.

▶ **Corollary 28.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. There exists a constant c such that the following are equivalent for all $0 < \epsilon < 1$:*

1. *There exists a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against learn-and-evaluate protocols.*
2. *There exists a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

Proof. We start with the $1 \implies 2$ implication. If 1 holds, then by Theorem 27 it follows that $\mathsf{prAM} \subseteq \mathcal{C}$. This in turn implies 2 by Theorem 11. As for the other direction, if a function f is not (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against learn-and-evaluate protocols, then it is also not (n^c, n^ϵ)-leakage-resilient hard against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ since a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ algorithm can use the SAT oracle to determine the output of the evaluation protocol. ◀

B Connection to non-uniform lower bounds

In this appendix, we prove Theorem 7, which establishes further equivalences between whitebox and blackbox mild derandomization of prAM. In particular, we show that uniform leakage-resilient hardness assumptions imply the $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ separation.

We start by presenting the equivalence between lower bounds for $\Sigma_2\mathsf{E}$ against nondeterministic circuits and against deterministic circuits with non-adaptive oracle access to SAT. Since we only need this equivalence in the case of polynomial-size circuit lower bounds, we state it in this setting.

▶ **Lemma 29** (Instantiation of [19, Theorem 3.2]). $\Sigma_2\mathsf{E} \subseteq \mathsf{NP/poly}$ *if and only if* $\Sigma_2\mathsf{E} \subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$.

Proof. The direction $\Sigma_2\mathsf{E} \subseteq \mathsf{NP/poly} \implies \Sigma_2\mathsf{E} \subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ is trivial, so we focus on the converse implication. Theorem 3.2 in [19] shows that if a low-degree extension $\hat{g}$ with specific parameters of a function g has non-adaptive SAT-oracle circuits of size s, then g itself has nondeterministic circuits of size $\mathsf{poly}(s)$. Assume that $\Sigma_2\mathsf{E} \subset \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$. By a standard argument, $\Sigma_2\mathsf{E}/n \subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ and there exists a constant c such that every language in $\Sigma_2\mathsf{E}/n$ has non-adaptive SAT-oracle circuits of size n^c.

Let $L \in \Sigma_2\mathsf{E}$. By having as advice n bits describing the number of strings of length n in L, a Σ_2-machine can compute the characteristic function $g_L : \{0,1\}^n \to \{0,1\}$ of L by guessing which strings of length n are in L and verifying each one in $\Sigma_2\mathsf{E}$, outputting 1 if and only if the input string is in the list of guessed-and-verified strings. We then set parameters exactly as in [19], that is, for a parameter $r' = 2(n + \log(32n^{5c}))$, we set $h = (4r')^2(9n^c)^4$, $r = n/\log h + 3$ and p to the smallest prime greater than or equal to $9hdr'$ to obtain the low-degree extension $\hat{g}$ of g. With these parameters, it follows that the function $\hat{g}_{\text{bool}}$ that maps the binary representation of a $\vec{y} \in \mathbb{F}_p^r$ and an index $i \in [\log p]$ to the i-th bit of $\hat{g}(\vec{y})$ is in $\Sigma_2\mathsf{E}/n$. Together with the assumption on $\Sigma_2\mathsf{E}$, we have that $\hat{g}$ has non-adaptive SAT-oracle circuits of size $s(n) = O(n^{c+1})$, and thus Theorem 3.2 in [19] guarantees that g (and thus L) has nondeterministic circuits of size $\mathsf{poly}(s(n)) = \mathsf{poly}(n)$. ◀

Now, we provide some intuition for the proof of Theorem 7. The first step is to understand how lower bounds such as $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ imply leakage-resilient hardness. First, by Lemma 29, the assumption implies also that $\Sigma_2\mathsf{E} \not\subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$. Define a function f that maps every input x of length ℓ to the truth-table of a hard language $L \in \Sigma_2\mathsf{E}$. With ℓ bits of advice (indicating how many strings of length ℓ are in L), this function can be computed in Σ_2-time $2^{O(\ell)}$. Assume, with the intent of deriving a contradiction, that f is not locally leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. For each input x where hardness fails, there exists a small leakage string a such that $A(x,a)$ locally computes f. That is, $L \in \mathsf{BPP}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ and thus also in $\mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ by Adleman's argument [1]. As for the other direction, by using the techniques developed in Section 4, we show that leakage-resilient hardness is sufficient for obtaining the mild derandomization of item 1, and thus equivalent to non-uniform lower bounds in the low end by [2].

We now detail how we prove Theorem 7. It is already known that items 1 and 2 are equivalent by the main result of [2]. We show that $2 \implies 3$ in Lemma 30, that $4 \implies 1$ in Lemma 31 and that $1 \iff 5$ in Lemma 33. The $3 \implies 4$ implication is trivial.

We start by obtaining leakage-resilient hardness from the assumption $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$.

▶ **Lemma 30.** *Assume* $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$. *Then for all* $\epsilon > 0$ *there exists a relation* $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ *that is* $\mathsf{poly}(n)$*-local* $(\infty, \mathsf{poly}(n))$*-leakage-resilient hard on all inputs of infinitely-many input lengths against* $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.

Proof. Let $L \in \Sigma_2\mathsf{E}$ and fix $\epsilon > 0$. We construct a function f from L such that, if f is not hard as in the theorem statement, then $L \in \mathsf{NP/poly}$. The existence of a hard relation then follows. Define f as the function that maps any input $x \in \{0,1\}^n$ to the truth-table of L at input length $n^{\epsilon/2}$. By having as advice the number N of strings of length $n^{\epsilon/2}$ in L, it is possible to compute this function in Σ_2-time $2^{O(n^{\epsilon/2})} \leq 2^{n^\epsilon}$ for sufficiently large n by guessing which N of the $2^{n^{\epsilon/2}}$ strings of length $n^{\epsilon/2}$ are in L and verifying those using the linear-exponential time Σ_2-algorithm for L. Note that since $f(x)$ is constant for all inputs of a given length, the same advice string applies to all such inputs.

Now, assume that f is not $\mathsf{poly}(n)$-local $(\infty, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. This means there exist a constant c, a (potentially uncomputable) function Leak and a probabilistic algorithm A with non-adaptive oracle access to SAT such that for almost-all input lengths n there exists $x \in \{0,1\}^n$ such that $|\mathrm{Leak}(x, f(x))| \leq n^c$ and $A(x, \mathrm{Leak}(x, f(x)))$ computes $f(x)$ locally in time n^c. By providing x and a "good" leakage string a as advice, algorithm $A(x, a)$ computes f locally in time n^c, and thus computes L at input length $n^{\epsilon/2}$ in time $\mathsf{poly}(n)$. This implies that $L \in \mathsf{BPP}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ and thus $L \in \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ [1]. By Lemma 29, $L \in \mathsf{NP}/\mathsf{poly}$. ◀

Now, we show that a weaker hardness assumption, where the leakage-providing function is computable in subexponential time, suffices to derandomize prAM.

▶ **Lemma 31.** *If for all $\epsilon > 0$ there exists a relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is $\mathsf{poly}(n)$-local $(2^{n^\epsilon}, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, then $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$.*

Proof. The proof is almost identical to that of Theorem 15. Fix some $\epsilon > 0$, the idea to construct a targeted hitting-set generator is to, on input $x \in \{0,1\}^{m'}$ representing a circuit D_x of size m for $m' = \Theta(m \log m)$, guess-and-verify a value $y \in R(x)$ and instantiate the generator H_{det} of Lemma 13 with y and m. The process takes time $2^{O(m^\epsilon)}$ and requires $O(m^\epsilon)$ bits of advice (for computing $y \in R(x)$). As with Theorem 15, there exist probabilistic algorithms Leak and A with non-adaptive oracle access to SAT such that for any $x \in \{0,1\}^n$ representing a circuit D_x not hit by the generator there exists $y \in R(x)$ for which the following holds. On input (x, y), Leak runs in time $2^{O(n^\epsilon)}$ and produces $\mathsf{poly}(n)$ bits of leakage that, when given as input to A, allow it to locally compute y in time $\mathsf{poly}(n, \log(2^{n^\epsilon})) = \mathsf{poly}(n)$.

As the hardness assumption holds for every input of infinitely-many input lengths (and R is total on the same input lengths), the targeted generator also works for infinitely-many circuit sizes m, and thus the derandomization in the conclusion of the theorem follows along the lines of Proposition 9 together with padding. ◀

Finally, we show that the weak derandomization assumption of item 1 in Theorem 7 is equivalent to local leakage-resilient hardness against almost-maximal leakage. Before doing so, we present a variant of Proposition 20 for derandomizations with advice that only work for infinitely-many input lengths.

▶ **Proposition 32.** *Assume that $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$. Then $\mathsf{prBPP}_{||}^{\mathsf{SAT}} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$.*

Proof (sketch). In [23, Lemma 38], it is shown that the hardness assumption $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP}/\mathsf{poly}$ implies that $\mathsf{prBPP}_{||}^{\mathsf{SAT}} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$ by using traditional hardness vs. randomness tradeoffs in the blackbox setting [22]. As the premise $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ implies that $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP}/\mathsf{poly}$, we are done. ◀

▶ **Lemma 33.** *The following are equivalent:*

1. $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ *for all* $\epsilon > 0$.
5. *For all $\epsilon > 0$ and $c \geq 1$ there exists a length-preserving relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is n^c-local $(n^c, \ell(n))$-leakage resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, where $n^{\Omega(1)} \leq \ell(n) \leq \omega(1)$ is polynomial-time computable.*

Proof. The proof follows closely the arguments of Theorems 21 (in the derandomization-to-hardness direction) and 15 (in the hardness-to-derandomization direction).

In the derandomization-to-hardness direction, recall that the proof of Theorem 21 shows that for every constant c, checking whether a value y is $(n^c, n - \omega(1))$ leakage-resilient hard for a string x (w.r.t. the first few algorithms) can be done in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. Such a value y must also be n^{c-2}-local leakage-resilient hard. Otherwise, we could amplify the algorithm A that locally computes y so that it outputs every bit of y correctly with high probability and obtain an algorithm that computes y in its entirety in time n^c. Fix some $\epsilon > 0$. By the derandomization assumption and Proposition 32, the check can be replaced by a $\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]$ algorithm with n^ϵ bits of advice that is guaranteed to work on infinitely-many input lengths. By guessing a value of y and running the Σ_2 verification, we obtain a relation R that is defined and hard for all inputs of infinitely-many input lengths.

The hardness-to-derandomization direction follows Theorem 15 even more closely. Theorem 15 establishes the hardness-to-targeted-derandomization connection on an instance-wise basis, i.e., the targeted generator hits every co-nondeterministic circuit D_x described by a padded string x' for which the hardness assumption holds. Since hardness holds for all inputs of infinitely-many input lengths, we obtain a targeted generator that works for infinitely many circuit sizes, which as in Lemma 31 suffices to obtain the derandomization result. ◀

We remark that due to the gap between the length of a solution $y \in R(x)$ and the leakage-resilient hardness in items 3 and 4 together with the sub-optimal scaling of the RMV reconstructor (Lemma 26) when compared to the SU reconstructor (Lemma 13), this result does not extend to hardness against learn-and-evaluate protocols.

Randomized and Quantum Query Complexities of Finding a King in a Tournament

Nikhil S. Mande
University of Liverpool, UK

Manaswi Paraashar
University of Copenhagen, Denmark

Nitin Saurabh
Indian Institute of Technology Hyderabad, India

Abstract

A tournament is a complete directed graph. It is well known that every tournament contains at least one vertex v such that every other vertex is reachable from v by a path of length at most 2. All such vertices v are called *kings* of the underlying tournament. Despite active recent research in the area, the best-known upper and lower bounds on the deterministic query complexity (with query access to directions of edges) of finding a king in a tournament on n vertices are from over 20 years ago, and the bounds do not match: the best-known lower bound is $\Omega(n^{4/3})$ and the best-known upper bound is $O(n^{3/2})$ [Shen, Sheng, Wu, SICOMP'03]. Our contribution is to show *tight* bounds (up to logarithmic factors) of $\widetilde{\Theta}(n)$ and $\widetilde{\Theta}(\sqrt{n})$ in the *randomized* and *quantum* query models, respectively. We also study the randomized and quantum query complexities of finding a maximum out-degree vertex in a tournament.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Quantum complexity theory

Keywords and phrases Query complexity, quantum computing, randomized query complexity, tournament solutions, search problems

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.30

Related Version *Full Version*: `https://arxiv.org/abs/2308.02472`

1 Introduction

A tournament is a complete directed graph. Many important properties of tournaments were studied by Landau [18] in the context of modelling dominance relations among a flock of chickens. Relevant to our paper is the notion of a *king* in a tournament. This notion was defined by Maurer [19], also with the goal of identifying a reasonable measure of dominance to identify a "most dominant" chicken in a flock. Soon after Maurer's article, Reid [22] showed existence of tournaments in which all vertices are kings. Tournaments also arise naturally in social choice theory where directions of edges depict preferences. A large amount of work has been devoted to defining a notion of a "winner" in a tournament, and determining the complexity of finding such winners. For instance, Dey [11] studied the complexity of certain tournament solutions with motivations from social choice theory. The monograph by Moon [20] sparked a line of research on tournaments and their structural properties.

A natural computational model to study the complexity of computing specific properties of a tournament, or more generally, a graph, is that of *query complexity*. In this setting an algorithm may query presence/directions of edges in an unknown input graph. The goal is to minimize the number of such queries made in the worst case. There is a rich literature on query complexity of graph problems, starting over 50 years ago [24, 23, 28, 15, 9, 12, 10, 11]. The famous Aanderaa-Karp-Rosenberg conjecture [24] or evasiveness conjecture posits that the query complexity of any non-trivial monotone graph property on n-vertex graphs has

© Nikhil S. Mande, Manaswi Paraashar, and Nitin Saurabh;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 30; pp. 30:1–30:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

maximal deterministic query complexity, i.e., $\binom{n}{2}$. While the deterministic and randomized variants of this conjecture are wide open, the quantum version was recently resolved in the positive [1] using Huang's breakthrough resolution of the sensitivity conjecture [16].

Our work deals with the query complexity of certain graph problems. In the next section, we describe the main graph problem of interest to us, and prior work on it.

1.1 Related work

It is well known that every tournament has at least one vertex v such that every other vertex is reachable from v via a path of length at most 2 (see Lemma 4 for a proof). Such a vertex v is called a *king* in the underlying tournament. Formally, one may define the following relation that captures this definition.

▶ **Definition 1** (Kings in a tournament). *Let T be a complete directed graph on n vertices. Identify the orientation of the tournament with a string T in $\{0,1\}^{\binom{n}{2}}$, one variable ($\{i,j\}$ with $i \neq j \in [n]$) per edge (between vertex i and vertex j) defining its direction. Define the relation $\mathsf{KING}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$ by*

$$(T, v) \in \mathsf{KING}_n \text{ if } \forall u \in [n], \text{ either } v \to u \text{ or } \exists w \text{ such that } v \to w \to u.$$

Here the directions of the edges $v \to u$ and $v \to w \to u$ are as in T.

A natural question arises: what is the query complexity of finding a king in an n-vertex tournament? The study of this was initiated by Shen, Sheng and Wu [25]. They showed an algorithm with query complexity $O(n^{3/2})$ and also showed a non-matching lower bound of $\Omega(n^{4/3})$. For the upper bound, they crucially used the fact that a king in an in-neighbourhood of an arbitrary subset of vertices is also a king in the original tournament (see Lemma 5). An outline of their upper bound is as follows: first arbitrarily choose a sub-tournament of a fixed size and find the maximum-out-degree vertex in it by querying all edges in this sub-tournament. Remove this vertex along with its out-neighbours, and proceed iteratively. When the number of remaining vertices is small enough, find a king using brute force (query all the edges in the remaining sub-tournament). Simple manipulation of parameters gives an upper bound of $O(n^{3/2})$. For the lower bound they design an adversary who answers an algorithm's queries using a fixed strategy, and show that every algorithm must make $\Omega(n^{4/3})$ queries in the worst case. Ajtai et al. [3] independently showed the same bounds, in a different context. Despite active recent research in the area (see the next paragraph), these bounds from over 20 years ago remain state-of-the-art. It can be shown that a vertex with maximum out-degree is a king (see Lemma 4 and its proof). However, finding a vertex with maximum out-degree is known to be hard: it has deterministic query complexity $\Omega(n^2)$ [4].

Biswas et al. [6] recently showed that the adversary used by [3, 25] to show an $\Omega(n^{4/3})$ lower bound cannot be used to prove a stronger lower bound. They additionally showed a query complexity upper bound of $O(n^{4/3})$ on finding a vertex from which at least half of all vertices are reachable by paths of length at most 2. They also considered variants of kings, and the complexity of finding such vertices. In a more recent work, Lachish, Reidl and Trehan [17] showed an $O(n^{4/3})$-query algorithm to find a vertex from which at least $(\frac{1}{2} + \frac{2}{17})$ of the vertices are reachable by paths of length at most 2.

1.2 Our contributions

While the question of pinning down the deterministic query complexity of finding a king has been open and unimproved since the work of Shen, Sheng and Wu [25], the corresponding question in the randomized and quantum query models does not seem to have been studied in

the literature. Our contribution is to give tight bounds in these models. We refer the reader to Section 2 for a formal description of these models. Let $\mathsf{R}(\cdot)$ and $\mathsf{Q}(\cdot)$ denote bounded-error randomized and quantum query complexity, respectively. Our main theorems are as follows.

▶ **Theorem 2.** *For all positive integers n,*

$$\mathsf{R}(\mathsf{KING}_n) = O(n \log\log n), \qquad \mathsf{R}(\mathsf{KING}_n) = \Omega(n),$$
$$\mathsf{Q}(\mathsf{KING}_n) = O(\sqrt{n}\ \mathrm{polylog}(n)), \qquad \mathsf{Q}(\mathsf{KING}_n) = \Omega(\sqrt{n}).$$

We mentioned earlier that a vertex of maximum out-degree in a tournament is a king, and finding a vertex of maximum out-degree is known to have deterministic query complexity $\Omega(n^2)$. We show that even the randomized query complexity is $\Omega(n^2)$, and we also show bounds for the quantum query complexity of this task. Define the relation $\mathsf{MOD}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$ to consist of the elements (T, v) where v is a maximum out-degree vertex in the n-vertex tournament described by T.

▶ **Theorem 3.** *For all positive integers n,*

$$\mathsf{R}(\mathsf{MOD}_n) = \Theta(n^2), \qquad \mathsf{Q}(\mathsf{MOD}_n) = O(n^{3/2}), \qquad \mathsf{Q}(\mathsf{MOD}_n) = \Omega(n).$$

We suspect that $\mathsf{Q}(\mathsf{MOD}_n) = \Theta(n^{3/2})$, but we leave open the problem of closing the gap between the upper and lower bounds in the quantum setting.

Sketch of randomized upper bound for finding a king. As mentioned in Section 1.1, the upper bound of Shen, Sheng and Wu crucially uses the fact that a king in the in-neighbourhood of an arbitrary vertex is also a king in the original tournament (Lemma 5). A simple counting argument shows that a *uniformly random* vertex in an n-vertex tournament has out-degree $\Omega(n)$ with high probability. This suggests a natural randomized iterative algorithm: in each step sample a few vertices and query all edges incident on them, until a vertex with large out-degree in the current sub-tournament is seen. We then remove this vertex along with all its out-neighbours from the tournament, and iterate. Since a random vertex has out-degree that is linear in the number of vertices with high probability, this process results in a small sub-tournament (with at most $\sqrt{n}$ vertices) after $O(\log n)$ iterations. At this point we can afford to query the entire remaining sub-tournament to find a king in it, and it can be shown by applying Lemma 5 iteratively that this vertex is also a king the original tournament.

Sketch of quantum upper bound for finding a king. Our quantum algorithm follows the same structure as our randomized one, but we run into some issues during a naive simulation. The following are the issues, along with how we handle them:

- When trying to sample a vertex with high out-degree, we cannot afford to query all edges incident on a vertex to compute its out-degree since our algorithm needs to have query complexity essentially $O(\sqrt{n})$. To circumvent the need of querying all edges incident on a vertex to compute its in-degree, we use the subroutine of *approximate counting* [8] that returns an approximation of the in-degree but offers a quadratic speedup. It may seem like one could use a classical algorithm for approximate counting here, but such a classical sampling-based algorithm would require $\widetilde{\Omega}(n)$ queries if the number of in-neighbours is small, say $\mathrm{polylog}(n)$ (see the fourth bullet as to why such a case may arise).
- A second issue that arises is when we need to sample a vertex from the current sub-tournament. It is no longer clear how to do this in the quantum setting since we do not explicitly know the vertices remaining. However, we keep track of the set of vertices W

whose out-neighbours have effectively been removed in previous iterations. The number of iterations of the algorithm, and hence the size of W, is bounded by $O(\log n)$. We then use key properties of Grover's search algorithm: first set up a uniform superposition over all vertices. Next we "mark" the vertices in the current sub-tournament (call such vertices "good") using $O(\log n)$ queries: for a given vertex we only need to check if it is an out-neighbour of any of the vertices in W. Making these queries in superposition allows us to mark the good vertices in $O(\log n)$ queries. We then apply the Grover iterate a suitable number (at most $O(\sqrt{n})$) of times. At this point we make a measurement in the computational basis: by the correctness of Grover's algorithm, the probability of seeing a good vertex (i.e., a vertex in the current sub-tournament) is large. The structure of Grover's algorithm implies that conditioned on not seeing a bad vertex, each good vertex is seen with equal probability. This effectively simulates sampling a uniformly random vertex from the current sub-tournament.

- Having sampled a vertex v from the in-neigbourhood of W, the randomized algorithm next computes the out-degree of v in the in-neighbourhood of W. We cannot afford to do this exactly since we do not explicitly know the in-neighbourhood of W, and moreover it may be very large. We are able to get around this using similar ideas to that in the previous bullet.
- Finally, it is no longer clear how to do the final brute-force step in the last remaining sub-tournament since we do not explicitly know the remaining $\sqrt{n}$ vertices. To handle this, we first modify the randomized algorithm so as to only have $O(\text{polylog}(n))$ vertices remaining in this "brute-force" step, while still having only $O(\log n)$ iterations overall. Thus, the query complexity so far is still $\widetilde{O}(\sqrt{n})$. We can then use Grover's search repeatedly (or an improvement thereof, Theorem 11) to find all the remaining vertices with high probability in $\widetilde{O}(\sqrt{n})$ queries. At this point we can find a king in the remaining sub-tournament using $O(\text{polylog}(n))$ queries. By the same argument as in the randomized case, this vertex is also a king in the original tournament.

Sketch of lower bounds for finding a king. To show our lower bounds, we restrict our attention to a special class of tournaments, described below. Fix an arbitrary tournament T on n vertices where vertex n is a source. This immediately implies that vertex n is the unique king. For each $i \in [n-1]$, define the tournament T_i to be T with edges incident on vertex i flipped so as to make vertex i the source. Note that these sets of edges are disjoint for every $i \neq j$. If we assign one variable to each such set and promise that at most one of them has value 1 (i.e., has edges in the opposite direction from those in T), an algorithm that finds a king in these tournaments (which are unique by construction) also solves the Search problem on $n-1$ input variables. Our lower bounds of $\Omega(n)$ and $\Omega(\sqrt{n})$ on the randomized and quantum query complexities, respectively, then immediately follow from corresponding well-known lower bounds on the complexity of the Search problem.

Sketch of bounds for finding a maximum out-degree vertex. In the randomized setting, we use Yao's minimax principle (Lemma 21). By this principle, it suffices to exhibit a hard distribution on input tournaments such that any deterministic algorithm with small query complexity must make large error when inputs are drawn from this distribution. We now describe the distribution: fix an n-vertex regular tournament, say T with n odd and each vertex having out-degree exactly $(n-1)/2$ (such a tournament is easy to construct iteratively, for example) and flip a uniformly random edge of T. This causes a unique vertex of the new tournament to be a maximum out-degree vertex. Intuitively, finding this vertex is as

hard as finding the edge that has been flipped, and it is well known that searching for a marked element among k elements has randomized query complexity $\Omega(k)$. We formalize this argument in Theorem 20. Our quantum lower bound uses similar ideas, and involves a reduction from the Search problem on an $\binom{n}{2}$-bit string, which has quantum query complexity $\Omega(n)$ [7]. For the quantum upper bound, we use a maximum finding routine over the degrees of the vertices. Each degree can be computed using $n-1$ queries, and the maximum can be found in $O(\sqrt{n})$ queries [13], giving us an $O(n^{3/2})$ upper bound.

2 Preliminaries

All logarithms in this paper are base 2. We use the notation $\mathrm{polylog}(n)$ to denote a quantity that is $\log^c n$ for a constant $c > 0$ (independent of n). For a positive integer n, we use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For an event X, let $\mathbb{I}[X]$ denote the indicator of X, i.e., $\mathbb{I}[X] = 1$ if X occurs, and $\mathbb{I}[X] = 0$ if X does not occur.

2.1 Tournaments

A tournament T on a vertex set V is a complete graph such that each edge is directed. Throughout this paper, unless mentioned otherwise, we consider tournaments T on n vertices and denote the vertex set by $V = [n]$. Such a tournament has $\binom{n}{2}$ directed edges. We identify an n-vertex tournament with a binary string in $\{0,1\}^{\binom{n}{2}}$: an element of $[n]$ corresponds to the label of a vertex, and there is one variable ($\{i,j\}$ with $i \neq j \in [n]$) per edge (between vertex i and vertex j) that defines its direction. For a tournament T and vertex $v \in V$, let $N^-(v)$ denote the set of in-neighbours of v, i.e., $N^-(v) = \{u \in [n] \setminus \{v\} \,|\, u \to v \text{ is an edge in } T\}$ and let $N^+(v)$ denote the set of out-neighbours of v (i.e., $\{u \in [n] \setminus \{v\} \,|\, v \to u \text{ is an edge}\}$). Also, let $d^+(v) = |N^+(v)|$ and $d^-(v) = |N^-(v)|$ denote the out-degree and in-degree of v, respectively. Since T is a tournament, $d^+(v) + d^-(v) = (n-1)$ for all $v \in V$. For $S \subseteq V$, let $T[S]$ be the tournament induced on the vertices in S. For a subset $W \subseteq V$, define $W^- = \{v \in V \mid v \to w \text{ is an edge for all } w \in W\}$. If $W = \emptyset$ then define $W^- = V$. A vertex $v \in V$ is a *king* if every vertex in $V \setminus \{v\}$ is reachable from v by a path of length at most 2. This is formally captured in Definition 1 and repeated below for convenience. Define the relation $\mathsf{KING}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$ by $(G, v) \in \mathsf{KING}_n$ if $\forall u \in [n] \setminus \{v\}$, either $v \to u$ or $\exists w : v \to w \to u$. Here the directions of the edges $v \to u$ and $v \to w \to u$ are as in the tournament G. A well-known fact about tournaments is that every tournament has a king. We give a proof for completeness.

▶ **Lemma 4** (Folklore). *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament. Then there exists a vertex $v \in [n]$ such that $(T, v) \in \mathsf{KING}_n$.*

Proof. Consider a vertex v of maximum out-degree. We show that such a vertex is a king. Consider the partition of V into three disjoint sets: $\{v\}$, $N^+(v)$ and $N^-(v)$. Clearly, every vertex in $N^+(v)$ is at a distance at 1 from v. Towards a contradiction, assume that there is a vertex w in $N^-(v)$ such that there is no path of length 2 of the form $v \to u \to w$, for some $u \in N^+(v)$. Thus every vertex in $N^+(v)$ is an out-neighbour of w. Since v is also an out-neighbour of w, the out-degree of w is greater than that of v, which is a contradiction. ◀

The above lemmas shows that any vertex with maximum out-degree in a tournament is a king in that tournament. However, as discussed in Section 1.1, finding a vertex of maximum out-degree is known to be hard. We need the following result due to [19].

▶ **Lemma 5** ([19]). *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament and $v \in [n]$. If a vertex u in $N^-(v)$ is a king in $T[N^-(v)]$, then u is a king in T.*

The proof of the above lemma is easy: If u is a king of the tournament $T[N^-(v)]$, then every vertex in $N^-(v)$ is at a distance at most 2 from u. Also, since u is an in-neighbour of v, every vertex in $N^+(v)$ is at a distance 2 from u. We also need the following lemma from [19].

▶ **Lemma 6** ([19]). *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament. $\sum_{i=1}^{n} d^+(i) = \sum_{i=1}^{n} d^-(i) = \binom{n}{2}$.*

We also need the following observation on the structure of a tournament (see e.g., [4]).

▶ **Lemma 7.** *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament and $k \geq 0$. Then, the number of vertices v such that $d^+(v) \leq k$ is at most $2k+1$.*

2.2 Query complexity

A deterministic decision tree T on m variables is a binary tree where the internal nodes are labeled by variables and leaves are labeled with elements of a set $\mathcal{R}$. Each internal node has a left child, corresponding to an edge labeled 0, and a right child corresponding to an edge labeled 1. On an input $x \in \{0,1\}^m$, T's computation traverses a path from root to leaf as follows. At an internal node, the variable associated with that node is *queried*: if the value obtained is 0, the computation moves to the left child, otherwise it moves to the right child. The output of T on input x, denoted by $T(x)$, is the label of leaf node reached. We say that a decision tree T computes the relation $f \subseteq \{0,1\}^m \times \mathcal{R}$ if $(x, T(x)) \in \mathcal{R}$ for all $x \in \{0,1\}^m$. The deterministic query complexity of f, is $\mathsf{D}(f) := \min_{T:T \text{ computes } f} \operatorname{depth}(T)$. A randomized decision tree $\mathcal{A}$ is a distribution $\mathcal{D}_{\mathcal{A}}$ over deterministic decision trees. On input $x \in \{0,1\}^m$, the computation of $\mathcal{A}$ proceeds by first sampling a deterministic decision tree T according to $\mathcal{D}_{\mathcal{A}}$, and outputting the label of the leaf reached by T on x. We say $\mathcal{A}$ computes f with bounded error if for every input x, $\Pr[(x, \mathcal{A}(x)) \in \mathcal{R}] \geq 2/3$. The randomized query complexity of $f \subseteq \{0,1\}^m \times \mathcal{R}$ is defined as follows. $\mathcal{R}(f) = \min_{\substack{\mathcal{A} \text{ computing } f \\ \text{with error } \leq 1/3}} \max_{T:\mathcal{D}_{\mathcal{A}}(T)>0} \operatorname{depth}(T)$.

2.3 Preliminaries for quantum query complexity

We refer the reader to [21, 26] for basics of quantum computing. A quantum query algorithm $\mathcal{A}$ computing a relation $f \subseteq \{0,1\}^m \times \mathcal{R}$ begins in an input-independent initial state $|\psi_0\rangle$, applies a sequence of unitaries $U_0, O_x, U_1, O_x, \cdots, U_T$, and performs a measurement. Here, the unitaries $U_0, U_1, \ldots, U_T$ are independent of the input. The unitary operation O_x represents the "query" operation, and maps $|i\rangle|b\rangle$ to $|i\rangle|b \oplus x_i\rangle$ for all $i \in [m]$ and $|0\rangle$ to $|0\rangle$. We say that $\mathcal{A}$ is a bounded-error algorithm computing f if for all $x \in \{0,1\}^m$, the probability of outputting $b \in \mathcal{R}$ such that $(x, b) \in f$ is at least $2/3$. The bounded-error quantum query complexity of f, denoted by $\mathsf{Q}(f)$, is the least number of queries required for a quantum query algorithm to compute f with error at most $1/3$.

We also need some basic notions from Grover's search algorithm [14], a fundamental quantum algorithm, referring the reader to [26, Chapter 7] for more details. In the search problem, a quantum algorithm is given quantum query access to a string $x \in \{0,1\}^n$. It is convenient to work with the "phase-query" unitary $O_{x,\pm}$ which satisfies $O_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$. The goal is to find an $i \in [n]$ such that $x_i = 1$ with probability at least $2/3$ if such an i exists, otherwise return that there is no such element. An i which satisfies $x_i = 1$ is also called a *marked element* and thus the goal is to find a marked element with high probability, if such an element exists.

Let $t := |\{i \in [n] : x_i = 1\}|$. Grover's algorithm starts with the uniform superposition $|U\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle$, and proceeds by applying Grover's iterate (which is an application of $O_{x,\pm}$ followed by a reflection about $|U\rangle$) several times. After k applications of Grover's iterate the resulting state is

$$\sin((2k+1)\theta) \sum_{i:x_i=1} |i\rangle + \cos((2k+1)\theta) \sum_{i:x_i=0} |i\rangle, \tag{1}$$

where $\theta = \arcsin(\sqrt{t/n})$. It is known that Grover's algorithm finds a marked element in x (if it exists) with $O(\sqrt{n})$ applications of the query oracle $O_{x,\pm}$, and probability at least $2/3$. Standard error reduction yields the following theorem.

▶ **Theorem 8.** *Given query access to $x \in \{0,1\}^n$, there is a quantum algorithm that decides whether the Hamming weight of x is 0 or returns an $i \in [n]$ such that $x_i = 1$, with error at most δ. The query complexity of this algorithm is $O(\sqrt{n \cdot \log(1/\delta)})$.*

Grover's algorithm is known to be asymptotically optimal.

▶ **Theorem 9** ([7]). *A quantum algorithm that solves the Search problem with error $2/5$ on n-bit inputs must have query complexity $\Omega(\sqrt{n})$, even when the inputs are promised to have Hamming weight either 0 or 1.*

The following theorem, due to Dürr and Høyer [13], is a generalization of Grover's search algorithm, to find the maximum number in an input list.

▶ **Theorem 10** ([13]). *Let T be an unsorted table of n items. There exists a quantum query algorithm of cost $O(\sqrt{n})$ that has query access to T and returns the maximum element of T with probability at least $2/3$.*

We require the following theorem, essentially due to Boyer et al. [7].[1]

▶ **Theorem 11** ([7]). *Given query access to $x \in \{0,1\}^n$ with $|x| \geq k$, there is a quantum algorithm that outputs, with query complexity $O(\sqrt{(n/k)} \log(1/\delta))$ and error probability at most δ, an index $i \in [n]$ with $x_i = 1$.*

We obtain the following immediate corollary by repeating the algorithm in Theorem 11 k times and updating the "marked" elements after each application.

▶ **Corollary 12.** *Given an input parameter k and query access to $x \in \{0,1\}^n$, there is a quantum algorithm that does the following with query complexity $O(\sqrt{nk} \log\log(n))$ and error probability at most $1/\mathrm{polylog}(n)$:*

- *If $|x| \geq k$, it returns k distinct indices $i_1, \ldots, i_k \in [n]$ such that $x_{i_j} = 1$ for $j \in [k]$.*
- *If $|x| < k$, it outputs all indices i with $x_i = 1$, along with the information that $|x| < k$.*

Our quantum algorithm also uses quantum approximate counting as a sub-routine. Here, an algorithm is given query access to a string $x \in \{0,1\}^n$. The indices $i \in [n]$ such that $x_i = 1$ are again called "marked". For an input parameter ε the goal of the algorithm is to output a multiplicative $(1 \pm \varepsilon)$-approximation of the number of marked indices of x. An optimal quantum algorithm for approximate counting was first given by Brassard et al. [8]. We use a version due to Aaronson and Rall [2].

[1] Their bound is for bounded-error algorithms and does not have polylogarithmic factors in the query complexity. Standard error reduction gives us Theorem 11.

▶ **Theorem 13** ([2]). *There exists a quantum algorithm that, given $\varepsilon > 0$ and query access to a string $x \in \{0,1\}^n$, outputs an estimate $\widetilde{K}$ of $K = |\{i : x_i = 1\}|$ such that $K(1-\varepsilon) \leq \widetilde{K} \leq K(1+\varepsilon)$ with probability at least $(1-\delta)$. The query complexity of this algorithm is $O(\sqrt{n/K} \cdot 1/\varepsilon \cdot \log(1/\delta))$.*

3 Randomized algorithm

Throughout this section and the next, unless mentioned otherwise, a tournament T is assumed to be in $\{0,1\}^{\binom{n}{2}}$, and its vertex set is denoted by $V = [n]$. Query algorithms are assumed to have classical/quantum query access to the edge directions of T, that is, the individual bits of the corresponding $\binom{n}{2}$-bit string.

In this section we give a randomized algorithm for finding a king in a tournament $T \in \{0,1\}^{\binom{n}{2}}$ with query complexity $O(n \log\log n)$ and success probability at least $2/3$. First, we make the following simple observation, which shows that a randomly chosen vertex from $V = [n]$ has a large number of out-neighbours with high probability.

▶ **Lemma 14** (Out-degree of a random vertex is large). *For all positive integers n, a tournament $T \in \{0,1\}^{\binom{n}{2}}$ and a vertex $v \in V$ chosen uniformly at random, $d^+(v) \geq \lfloor (n-1)/5 \rfloor$ with probability at least $3/5$.*

Proof. From Lemma 7, $|\{v \in V \mid d^+(v) < \lfloor (n-1)/5 \rfloor\}| \leq 2((n-1)/5-1)+1 = (2n-7)/5 < 2n/5$. Thus, the fraction of vertices with out-degree at least $\lfloor (n-1)/5 \rfloor$ is at least $3/5$. ◀

Lemma 14 suggests a natural randomized query algorithm, given in Algorithm 1. We show in Theorem 15 that the algorithm makes $O(n \log\log n)$ queries to T in the worst case, and returns a king with probability at least $2/3$.

Algorithm 1 Randomized Query Algorithm.

Input: Query access to edge directions of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ where $V = [n]$.
while $|V| \geq \sqrt{n}$ **do**
  $t \leftarrow |V|$, $k \leftarrow \lceil \log\log n \rceil$
  $v_1, \ldots, v_k \leftarrow$vertices chosen uniformly at random from V
  $w \leftarrow \arg\max_{u \in \{v_1,\ldots,v_k\}} d^+(u)$ ▷ querying all edges incident on $\{v_1, \ldots, v_k\}$ in $T[V]$ and breaking ties arbitrarily
  if $d^+(w) = t-1$ **then**
    Return w
  else if $d^+(w) < \lfloor (t-1)/5 \rfloor$ **then**
    Return a random vertex $v \in V$
  else ▷ $\lfloor (t-1)/5 \rfloor \leq d^+(w) < t-1$ here
    $V \leftarrow N^-(w)$ ▷ This is the in-neighbourhood of w in the set V, and not in the whole vertex set $[n]$.
    continue
  end if
end while
$w \leftarrow$ a king in $T[V]$ ▷ query all edges in the sub-tournament $T[V]$
Output w

▶ **Theorem 15.** *Let $n > 0$ be a positive integer. Then,* $\mathsf{R}(\mathsf{KING}_n) = O(n \log\log n)$.

Proof. Consider Algorithm 1. We first analyze the query cost of the algorithm. For the correctness, we define "bad events", argue correctness of the algorithm conditioned on no bad event occurring, and then upper bound the probability of a bad event happening.

Query complexity. In order to upper bound the query complexity, first note that each iteration of the **while** loop (Line 2) uses $k \cdot |V| \leq |V| \log\log n$ queries in the worst case. Furthermore, the **while** loop goes into the next iteration (Line 12) if and only if $|V| > \sqrt{n}$ (Line 2) and a vertex w of out-degree at least $\lfloor (t-1)/5 \rfloor$ has been found in Line 5 (see comment on Line 10). This means that the size of the vertex set reduces by a factor of at least $4/5$ in the next iteration of the **while** loop. In particular, this means in the i'th iteration of the **while** loop, we have $|V| \leq (4/5)^i \cdot n$, and thus there are $O(\log n)$ iterations of the **while** loop in the worst case. Finally, Line 15 accounts for at most $O(n)$ queries since $|V| < \sqrt{n}$ here. The worst-case query complexity is thus upper bounded by

$$n + \sum_{i=0}^{O(\log n)} \left(\frac{4}{5}\right)^i \cdot n \cdot O(\log\log n) = O(n \log\log n).$$

Bad event, and correctness assuming no bad event. The event of Line 9 occurring during the run (i.e., Line 8 being triggered in any iteration) is defined to be the bad event. Conditioned on the bad event not occurring, the algorithm either terminates on Line 7 or Line 16. Clearly when the algorithm terminates on Line 7 or Line 16, the output vertex is a king in the sub-tournament being considered at the moment. If the **while** loop has not even completed once, the current sub-tournament is the same as the original tournament, and we are done. If the **while** loop has completed at least once, the sub-tournament being considered at the moment is the sub-tournament of a tournament T' (which itself may be a sub-tournament of T) induced by the in-neighbourhood of a specific vertex. Applying Lemma 5, we conclude that the king in the current sub-tournament is also a king in T', and also the whole tournament by applying Lemma 5 repeatedly now. Hence conditioned on the bad event not occurring, the algorithm indeed outputs a correct answer.

Probability of bad event. From Lemma 14, the probability that Line 8 is run in an iteration is at most $(2/5)^k \leq 1/\log^{\log 2.5} |V| \leq 1/\log^{1.3} n$. By a union bound, the probability that Line 8 gets executed in any of the $O(\log n)$ iterations is at most $O(\log n)/\log^{1.3}(n) = o(1)$. ◀

4 Quantum algorithm

For $W \subseteq [n]$ and $v \in V$, we can decide whether v is an out-neighbour of any $w \in W$ by making $|W|$ queries, by checking x_{wv} for all $w \in W$. Similarly, $|W|$ queries are sufficient to decide whether v is an in-neighbour of some vertex $w \in W$. This simple classical algorithm can easily be simulated in the quantum setting, which gives us the following observation.

▶ **Observation 16.** *For a tournament $T \in \{0,1\}^{\binom{n}{2}}$ and a known subset of the vertices $W \subseteq V$, there exists a unitary transformation that maps the basis state $|v\rangle$ to $(-1)^{\mathbb{I}[v \in W^-]}|v\rangle$ using $|W|$ queries to T. In other words, there is a unitary transformation that has query cost $|W|$ and "marks" vertices in W^-.*

Before proving the main theorem of this section, we give two lemmas (proven in the appendix). The algorithm in these lemmas will be used in the proof of the main theorem.

▶ **Lemma 17.** *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament, $W \subseteq V$ and $t = \Theta(\log\log n)$ be an integer. There exists a quantum algorithm* In-Sample(T, W, t)*, Algorithm 2, that with error probability at most $1/(\text{polylog}(n))$, returns a set of uniformly distributed and independent samples from W^- of size t. The query complexity of this algorithm is $O(|W| \cdot \sqrt{n} \cdot \text{polyloglog}(n))$.*

Algorithm 2 The In-Sample(T, W, t) algorithm for sampling many uniformly independent samples from a subset of vertices.

1: **Input:** Query access to the adjacency matrix of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ where $V = [n]$, $W \subseteq V$ such that $|W^-| \geq \log^{100} n$, and $t \in \mathbb{N}$ such that $t = \Theta(\log\log n)$.
2: $N \leftarrow 10^4 n$
3: $|\phi\rangle \leftarrow \sum_{i=1}^{N} \frac{1}{\sqrt{N}} |i\rangle$ ▷ `$|\phi\rangle$ is used as the starting state in Line 8 and Line 14 with vertices in $W^- \subseteq [n]$ marked (by first checking if $j \in [N]$ satisfies $j \leq n$, and marking such a $j$ using $|W|$ queries).`
4: **if** $W = \emptyset$ **then**
5: $S \leftarrow t$ samples from uniform superposition over V
6: Return S
7: **else**
8: $\widetilde{w} \leftarrow$ estimate of $|W^-|$ from Theorem 13 with $\varepsilon = 1/100, \delta = 1/\text{polylog}(N) = 1/\text{polylog}(n)$.
9: $w' \leftarrow \lfloor \widetilde{w}/2 \rfloor$
10: $\widetilde{k} \leftarrow \left\lfloor \left(\frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \right\rfloor$
11: $R \leftarrow \emptyset$
12: $\mathsf{count} \leftarrow 0$
13: **while** $\mathsf{count} < O(t \text{ polyloglog}(n))$ **do**
14: $|\psi_i\rangle \leftarrow$ state obtained by applying Grover's iterate $\widetilde{k}$ times on $|\phi\rangle$, with vertices in W^- being the marked elements
15: $v_i \leftarrow$ measurement outcome of $|\psi_i\rangle$ in computational basis
16: **if** $v_i \in W^-$ **then** ▷ `query edges between $v_i$ and $W$`
17: $R \leftarrow R \cup \{v_i\}$
18: **end if**
19: $\mathsf{count} \leftarrow \mathsf{count} + 1$
20: **if** $|R| = t$ **then** ▷ `If we have collected enough samples`
21: Return R ▷ `This is a set of uniformly distributed and independent samples from $W^-$ of size $t$ (See Lemma 17)`
22: **end if**
23: **end while**
24: **end if**
25: Return $[t]$ ▷ `The algo makes error in this case.`

▶ **Lemma 18.** *Let $T \in \{0,1\}^{\binom{n}{2}}$ be a tournament, W be a subset of V satisfying $|W^-| \geq \log^{100} n$ and u be a vertex in V. There exists a quantum algorithm* Decide-High-Out-Degree(T, W, u)*, Algorithm 3, that returns with error probability at most $1/(\text{polylog}(n))$,* True *if the out-degree of u in W^- is at least $|W^-|/5$ and* False *if the out-degree of u in W^- is at most $|W^-|/10$. The query complexity of this algorithm is $O(|W| \cdot \sqrt{n}\ \text{polylog}(n))$.*

Algorithm 3 The Decide-High-Out-Degree(T, W, u) subroutine.

1: **Input:** Query access to the edge directions of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ where $V = [n]$, $W \subseteq V$ such that $|W^-| \geq \log^{100} n$, and $u \in V$.
2: $\widetilde{w}_1 \leftarrow$ estimate of $|W^-|$ using Theorem 13 with $\varepsilon = 1/100, \delta = 1/\text{polylog}(n)$.
▷ Since the algorithm is given W is input, it can decide whether $v \in W^-$ by making $|W|$ queries.
3: $\widetilde{w}_2 \leftarrow$ estimate of $|N^+(u) \cap W^-|$ using Theorem 13 with $\varepsilon = 1/100, \delta = 1/\text{polylog}(n)$.
▷ Note that we do not have query access to the presence/absence of a vertex v in $N^+(u) \cap W^-$. However such a query can be implemented with $1 + |W|$ queries: check if $v \to u$ is an edge, and check if $v \to w$ is an edge for any $w \in W$.
4: **if** $\widetilde{w}_2/\widetilde{w}_1 \geq 99/505$ **then**
5: return True
6: **else**
7: return False
8: **end if**

We now show our main result of this section.

▶ **Theorem 19.** *Let $n > 0$ be a positive integer. Then* $\mathsf{Q}(\mathsf{KING}_n) = O(\sqrt{n}\ \text{polylog}(n))$.

Proof. Consider Algorithm 4. We first analyze the query cost of the algorithm. For the correctness, we define "bad events", argue correctness of the algorithm conditioned on no bad event occurring, and then upper bound the probability of a bad event happening.

Query complexity. First we upper bound $|W|$ at the end of the run of the algorithm. The **while** loop in Line 3 runs for at most $O(\log n)$ iterations. The algorithm starts with W initialized to $\emptyset$ and is updated only in Line 14 where one new element is added to W. Thus we have $|W| = O(\log n)$.

Consider Line 5. Since $|W| = O(\log n)$ and $k = \log^{100} n$, by Corollary 12 the number of queries in this step is upper bounded by $O(|W|\sqrt{n}\ \text{polylog}(n)) = O(\sqrt{n}\ \text{polylog}(n))$, and thus the overall cost of queries executed in this line over at most $O(\log n)$ iterations is also $O(\sqrt{n}\ \text{polylog}(n))$.

In Line 9, the In-Sample algorithm (Algorithm 2) is called at most $O(\log n)$ times with $t = \Theta(\log\log n)$ and $|W| = O(\log n)$. Thus by Lemma 17, the cost of this step is upper bounded by $O(|W|\sqrt{n}\ \text{polylog}(n)) = O(\sqrt{n}\ \text{polylog}(n))$.

Now consider the **for** loop in Line 11. This loop is executed at most $O(\log n)$ times and each iteration of this loop invokes the algorithm Decide-High-Out-Degree, with $|W| = O(\log n)$, at most $|S|$ many times. Since $|S| = O(\text{polylog}(n))$ (see Lemma 17) the query cost in this loop is upper bounded by $O(|W| \cdot |S| \cdot \sqrt{n}\ \text{polylog}(n)) = O(\sqrt{n}\ \text{polylog}(n))$ in the worst case.

The only remaining step in Line 23. In this case, since $|U| \leq \log^{100} n$ throughout the algorithm, at most $O(\text{polylog}(n))$ queries are made.

Bad event, and correctness assuming no bad event. If any of the following events happen, we say that a bad event has happened for Algorithm 4:

(I) The algorithm in Corollary 12 which is used in Line 5 gives an incorrect answer.
(II) The algorithm In-Sample (Algorithm 2) in Line 9 fails to return a set of $\Omega(t) = \Omega(\log\log n)$ uniformly distributed and independent samples from W^-.
(III) The set S obtained from In-Sample in Line 9 does not contain a vertex of out-degree at least $|W^-|/5$ in W^-.
(IV) The algorithm Decide-High-Out-Degree (Algorithm 3) in Line 13 returns False.

We prove the correctness of the algorithm assuming that these bad events do not happen. Consider the j'th iteration of the **while** loop in Line 3, for $j \geq 1$, and let $W^{(j)}$ denote the set W in this iteration. $W^{(j)}$ is updated only in Line 14 by a v which satisfies $v \in (W^{(j)})^-$. This is because each vertex of the set S belongs to W^- (see Line 16 of Algorithm 2). In the next iteration of the **while** loop, $(W^{(j+1)})^-$ is defined as $(W^{(j)})^- \cap N^-(v)$. Thus by applying Lemma 5 iteratively, $(W^{(j+1)})^-$ contains a king in the tournament $T[(W^{(j)})^-]$, and hence a king in T.

Assuming that the bad events do not happen, we now argue that in $O(\log n)$ iterations the size of W^- becomes smaller than $\log^{100} n$. In this case $U = W^-$ because of the property of Corollary 12 used in Line 5, and the algorithm correctly returns the king in Line 23 by a similar argument as in the previous paragraph by iteratively applying Lemma 5. The analysis is similar to that of proof of Theorem 15. Since Decide-High-Out-Degree (Algorithm 3) in Line 13 does not return False, the out-degree of v in $(W^{(j)})^-$ must be at least $|(W^{(j)})^-|/10$.

Thus $|(W^{(j+1)})^-| \leq (9/10) \cdot |(W^{(j)})^-|$, and after $O(\log n)$ iterations the size of W^- is smaller than $\log n < \log^{100} n$.

Probability of bad event. The probability of events I, II, IV are each upper bounded by $O(1/\text{polylog}(n))$ by Corollary 12, Lemma 17 and Lemma 18, respectively. The probability of event III conditioned on II not happening is upper bounded by $(2/5)^{\Theta(\log\log(n))} = O(1/\text{polylog}(n))$, thus the probability of event III is upper bounded by $O(1/\text{polylog}(n))$. The number of times that the events I, II, III can happen is at most $O(\log n)$, and IV can happen is at most $O(\text{polylog}(n))$, a union bound implies the probability of a bad event happening is upper bounded by $O(1/\text{polylog}(n))$. ◀

5 Lower bounds

We show our lower bounds in this section. We first show our lower bounds for the query complexity of finding a vertex of maximum out-degree, and then our lower bounds for finding a king in a tournament.

5.1 Maximum out-degree

We show in this subsection that the randomized query complexity of finding a vertex of maximum out-degree in an n-vertex tournament is $\Omega(n^2)$. This task is formally defined as the relation $\mathsf{MOD}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$: $(G, v) \in \mathsf{MOD}_n$ if $d^+(v) \geq d^+(w)\ \forall w \neq v \in [n]$. Here the out-degrees of v, w are according to the tournament G.

▶ **Theorem 20.** *For sufficiently large positive integers n, $\mathsf{R}(\mathsf{MOD}_n) \geq n^2/100$.*

We use Yao's minimax principle [27], stated below in a form convenient for us.

▶ **Lemma 21** (Yao's minimax principle). *For a relation $f \subseteq \{0,1\}^m \times \mathcal{R}$, we have $\mathsf{R}(f) \geq k$ if and only if there exists a distribution $\mu : \{0,1\}^m \to [0,1]$ such that $\mathsf{D}_\mu(f) \geq k$. Here, $\mathsf{D}_\mu(f)$ is the minimum depth of a deterministic decision tree that computes f to error at most $1/3$ when inputs are drawn from the distribution μ.*

Algorithm 4 Quantum Algorithm.

1: **Input:** Query access to the edge directions of a tournament $T \in \{0,1\}^{\binom{n}{2}}$ with $V = [n]$
2: $W \leftarrow \emptyset$, $t \leftarrow \Theta(\log\log n)$, and $\mathsf{COUNT} \leftarrow O(\log n)$
▷ Recall that $\emptyset^- := V$
3: **while** $\mathsf{COUNT} > 0$ **do**
4: $\quad \mathsf{COUNT} \leftarrow \mathsf{COUNT} - 1$
5: $\quad U \leftarrow$ the output of the algorithm in Corollary 12 with the string in $\{0,1\}^{[n]}$ as input where indices corresponding to vertices in W^- are equal to 1 (marked), and $k = \log^{100} n$
▷ query access to this string can be done using $|W|$ edge queries to T
6: $\quad$ **if** $|U| < \log^{100} n$ **then**
7: $\quad\quad$ **break** ▷ Go to Line 23
8: $\quad$ **else**
9: $\quad\quad S \leftarrow$ In-Sample(T, W, t) ▷ We reach here if $|W^-| \geq \log^{100} n$ (Line 7 gets executed otherwise)
10: $\quad\quad S' \leftarrow S$
11: $\quad\quad$ **for** $v \in S$ **do**
12: $\quad\quad\quad S' \leftarrow S' \setminus \{v\}$
13: $\quad\quad\quad$ **if** Decide-High-Out-Degree$(T, W, v) == \mathsf{True}$ **then**
▷ Decide-High-Out-Degree can be applied since $|W^-| \geq \log^{100} n$
14: $\quad\quad\quad\quad W \leftarrow W \cup \{v\}$
15: $\quad\quad\quad\quad$ **break** ▷ Go to Line 3
16: $\quad\quad\quad$ **end if**
17: $\quad\quad\quad$ **if** $S' == \emptyset$ **then**
18: $\quad\quad\quad\quad$ Return a random vertex $v \in V$
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end for**
21: $\quad$ **end if**
22: **end while**
23: Return a king in U ▷ query all edges in $T[U]$

Proof of Theorem 20. Assume without loss of generality that n is odd. We construct a hard distribution μ on n-vertex tournaments. We show that any deterministic query algorithm of cost less than $n^2/100$ must make error at least $1/3$ on inputs drawn from μ, and this would prove the theorem by Yao's principle (Lemma 21). Let G be a fixed n-vertex regular tournament where every vertex has out-degree exactly $(n-1)/2$ (such a tournament is easy to construct, by induction, for example). The distribution μ is defined by taking G and flipping the direction of a uniformly random edge. Note that all resultant tournaments have a unique vertex with maximum out-degree.

Consider a deterministic query algorithm (decision tree) that queries less than $n^2/100$ edges. Consider the leaf L of this tree for which answers of all queries on its path are consistent with directions of edges in G. Say the label of this leaf is vertex i. Consider the set S of all unqueried edges on the path to L that are not incident on vertex i. We have $|S| \geq \binom{n}{2} - \frac{n^2}{100} - (n-1)$. For each $e \in S$, the graph G_e defined by flipping the direction of e in G reaches the leaf L. Moreover, the unique maximum out-degree vertex of G_e is not vertex i since e is not incident on i by the definition of S. This implies that the tree outputs the wrong answer on G_e. By the definition of μ, we have $\mu(G_e) = 1/\binom{n}{2}$ for all $e \in S$. Thus, the mass of inputs under μ on which the decision tree makes an error is at least

$$\sum_{e \in S} \mu(G_e) \geq \frac{\binom{n}{2} - \frac{n^2}{100} - n + 1}{\binom{n}{2}} > \frac{97}{100} > \frac{1}{3},$$

where the second-to-last inequality holds for sufficiently large n. Lemma 21 yields the theorem. ◀

We now give our quantum bounds for MOD_n.

▶ **Theorem 22.** *For all positive integers n, $\mathsf{Q}(\mathsf{MOD}_n) = O(n^{3/2}), \mathsf{Q}(\mathsf{MOD}_n) = \Omega(n)$.*

Proof. For the upper bound we apply the maximum finding subroutine in Theorem 10 to the degree sequence of the input tournament. Finding the degree of a vertex (and hence a query of the maximum-finding algorithm) can be done with $n-1$ edge queries. Thus, this algorithm has cost $O(\sqrt{n} \cdot (n-1)) = O(n^{3/2})$.

For the lower bound, we give a reduction from the Search problem on an $\binom{n}{2}$-bit string. As in the proof of the randomized lower bound, assume n is odd and let G be a fixed n-vertex regular tournament where every vertex has out-degree exactly $(n-1)/2$. Towards a contradiction, suppose we have an algorithm $\mathcal{A}$ that finds a maximum out-degree vertex in an n-vertex graph with query complexity $o(n)$ and probability at least $2/3$. We use $\mathcal{A}$ to solve the Search problem on $\binom{n}{2}$-bit strings with the promise that the input has Hamming weight at most 1. On input $x \in \{0,1\}^{\binom{n}{2}}$ with $|x| \leq 1$, do the following:

1. Run the algorithm $\mathcal{A}$ on the tournament $G \oplus x$. Here $G \oplus x$ denotes the bitwise XOR of G and x. Suppose the output is $v \in [n]$.
2. Run a $(99/100)$-error Search algorithm with query cost $O(\sqrt{n})$ on the $n-1$ indices of x that are indexed by pairs with one element as v (that is, indexed by the edges adjacent to v in the corresponding tournament).
3. Output the index returned by the search algorithm.

The cost of this algorithm is clearly $o(n) + O(\sqrt{n})$. For the correctness, first note that when $|x| = 1$ and G is such that all out-degrees are equal, the tournament $G \oplus x$ has exactly one maximum out-degree vertex. Thus, by the correctness of $\mathcal{A}$, it outputs this vertex with probability at least $2/3$. Observe that the edge flipped in $G \oplus x$ from G is adjacent to this vertex. In the event that the first step outputs the correct vertex, the edge that has been flipped in $G \oplus x$ from G (i.e., the index $\{i,j\}$ with $x_{\{i,j\}} = 1$) is caught in the second step with probability at least $99/100$. Thus, this gives an algorithm solving the Search problem on $\binom{n}{2}$-bit strings with the promise that the input has Hamming weight at most 1, with success probability at least $(2/3) \cdot (99/100) > 3/5$. The query cost of this algorithm is $o(n)$ from the first step, by our assumption, and $O(\sqrt{n})$ from the second step. Thus the total cost is $o(n)$, which is a contradiction in view of Theorem 9. ◀

We leave open the question of closing the gap in Theorem 22.

5.2 Finding a king

We show an $\Omega(n)$ lower bound for the randomized query complexity of finding a king in a tournament, and an $\Omega(\sqrt{n})$ quantum query lower bound. To show these lower bounds, we restrict our attention on input tournaments of a particular structured form that have the property that there is only one king (which is a source in the tournament). We then show a lower bound on the randomized and quantum query complexities of finding a king in these promised inputs, by a reduction from the Search problem on $n-1$ variables with the promise

that the input has Hamming weight either 0 or 1, for which we know an $\Omega(n)$ lower bound in the randomized setting and an $\Omega(\sqrt{n})$ lower bound in the quantum setting. Our reductions use a simple modification of block sensitivity.

We require the following relation.

▶ **Definition 23.** *Let n be a positive integer. Define the relation* $\mathsf{USEARCH}_n \subseteq \{0,1\}^n \times \{\emptyset\} \cup [n]$ *as* $(0^n, \emptyset) \in \mathsf{USEARCH}_n$ *and* $(x, i) \in \mathsf{USEARCH}_n$ when $x = e_i$.

▷ Claim 24. Let n be a positive integer. Then,

$$\mathsf{R}(\mathsf{KING}_n) \geq \mathsf{R}(\mathsf{USEARCH}_{n-1}), \qquad \mathsf{Q}(\mathsf{KING}_n) \geq \mathsf{Q}(\mathsf{USEARCH}_{n-1}).$$

Proof. Consider an arbitrary input $x \in \{0,1\}^{\binom{n}{2}}$ such that the vertex n is the source. For each $j \in [n-1]$, let $V_j \subseteq \left[\binom{n}{2}\right]$ be the set of edges incident on vertex j that need to be flipped in the input x to make vertex j the source. We first make the following two observations:

$$V_j \cap V_k = \emptyset \quad \forall j \neq k \in [n-1], \qquad \bigcup_{i=1}^{n-1} V_j = \left[\binom{n}{2}\right]. \tag{2}$$

The first observation follows by considering an edge from vertex ℓ to vertex m. This edge only appears in V_m. Clearly every edge belongs to exactly one V_j, proving the second observation.

Using these two observations, the input set $\{0,1\}^{\binom{n}{2}}$ can also be expressed as $\{0,1\}^{V_1} \times \{0,1\}^{V_2} \times \cdots \times \{0,1\}^{V_{n-1}}$. For the remaining part of this proof we treat inputs to be of the latter form. In fact, we only restrict our attention to the case where each coordinate in a "block" has the same value.

For a string $y \in \{0,1\}^{n-1}$, define the tournament $x_y = \bigotimes_{i=1}^{n-1} y_i^{V_i}$. Thus we have the following tournaments when $|y| \leq 1$:

$$x_{e_j} = \begin{cases} 0^{V_1} \times \cdots \times 0^{V_{j-1}} \times 1^{V_j} \times 0^{V_{j+1}} \times \cdots \times 0^{V_{n-1}} & y = e_{j-1} \\ 0^{V_1} \times \cdots \times 0^{V_{n-1}} & y = 0^{n-1}. \end{cases}$$

In other words, x_{e_j} equals the tournament x with variables in V_j flipped, and $x_{0^{n-1}} = x$.

Note that vertex j is the source (and thus the unique king) in the tournament x_{e_j}. Thus, finding a king in the set of tournaments $\left\{x_{e_j} : j \in [n-1]\right\}$ is the same as finding a source in these tournaments. Thus, a query algorithm finding a king in the restricted input set $x_y : |y| \leq 1$ yields a query algorithm for $\mathsf{USEARCH}_{n-1}$ on input y, which proves the claim. ◁

From the well-known lower bounds of $\mathsf{Q}(\mathsf{USEARCH}_{n-1}) = \Omega(\sqrt{n})$ [5] and $\mathsf{R}(\mathsf{USEARCH}_{n-1}) = \Omega(n)$, we obtain our main theorem of this section.

▶ **Theorem 25.** *Let n be a positive integer and* $\mathsf{KING}_n \subseteq \{0,1\}^{\binom{n}{2}} \times [n]$. *Then,*

$$\mathsf{R}(\mathsf{KING}_n) = \Omega(n), \qquad \mathsf{Q}(\mathsf{KING}_n) = \Omega(\sqrt{n}).$$

References

1 Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1330–1342. ACM, 2021. doi:10.1145/3406325.3451047.

2 Scott Aaronson and Patrick Rall. Quantum approximate counting, simplified. In *Symposium on simplicity in algorithms*, pages 24–32, 2020.

3 Miklós Ajtai, Vitaly Feldman, Avinatan Hassidim, and Jelani Nelson. Sorting and selection with imprecise comparisons. *ACM Transactions on Algorithms (TALG)*, 12(2):1–19, 2015.

4 Ramachandran Balasubramanian, Venkatesh Raman, and G Srinivasaragavan. Finding scores in tournaments. *Journal of Algorithms*, 24(2):380–394, 1997.

5 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. doi:10.1137/S0097539796300933.

6 Arindam Biswas, Varunkumar Jayapaul, Venkatesh Raman, and Srinivasa Rao Satti. Finding kings in tournaments. *Discret. Appl. Math.*, 322:240–252, 2022. doi:10.1016/j.dam.2022.08.014.

7 Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.

8 Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

9 Amit Chakrabarti and Subhash Khot. Improved lower bounds on the randomized complexity of graph properties. In *Automata, Languages and Programming: 28th International Colloquium, ICALP 2001 Crete, Greece, July 8–12, 2001 Proceedings 28*, pages 285–296. Springer, 2001.

10 Andrew M Childs and Robin Kothari. Quantum query complexity of minor-closed graph properties. *SIAM Journal on Computing*, 41(6):1426–1450, 2012.

11 Palash Dey. Query complexity of tournament solutions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2992–2998. AAAI Press, 2017. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14180.

12 Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006.

13 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

14 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 212–219. ACM, 1996. doi:10.1145/237814.237866.

15 Péter Hajnal. An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11:131–143, 1991.

16 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.

17 Oded Lachish, Felix Reidl, and Chhaya Trehan. When you come at the king you best not miss. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022*, volume 250 of *LIPIcs*, pages 25:1–25:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.25.

18 HG Landau. On dominance relations and the structure of animal societies: Iii the condition for a score structure. *The bulletin of mathematical biophysics*, 15:143–148, 1953.

19 Stephen B Maurer. The king chicken theorems. *Mathematics Magazine*, 53(2):67–80, 1980.

20 John W Moon. *Topics on tournaments in graph theory.* Courier Dover Publications, 2015.

21 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition).* Cambridge University Press, 2016. URL: https://www.cambridge.org/de/academic/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-computation-and-quantum-information-10th-anniversary-edition?format=HB.

22 K.B. Reid. Every vertex a king. *Discrete Mathematics*, 38(1):93–98, 1982. doi:10.1016/0012-365X(82)90173-X.

23 Ronald L Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3(3):371–384, 1976.

24 Arnold L Rosenberg. On the time required to recognize properties of graphs: A problem. *ACM SIGACT News*, 5(4):15–16, 1973.

25 Jian Shen, Li Sheng, and Jie Wu. Searching for sorted sequences of kings in tournaments. *SIAM J. Comput.*, 32(5):1201–1209, 2003. doi:10.1137/S0097539702410053.

26 Ronald de Wolf. Quantum computing: Lecture notes, 2019. arXiv:1907.09415, version 5. arXiv:1907.09415.

27 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 222–227. IEEE Computer Society, 1977.

28 Andrew Chi-Chih Yao. Lower bounds to randomized algorithms for graph properties. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 393–400. IEEE, 1987.

A Proofs of Lemmas from Section 4

In this section we prove Lemma 17 and Lemma 18.

Proof of Lemma 17. Consider Algorithm 2. We first analyze the query cost of the algorithm. For the correctness, we define "bad events", argue correctness of the algorithm conditioned on no bad event occurring, and then upper bound the probability of a bad event happening.

Query complexity. We upper bound the worst-case query complexity of the algorithm. Line 8 of the algorithm case costs $O(|W| \cdot \sqrt{N} \cdot \text{polylog}(N))$ from Theorem 13. The **while** loop from Line 13 runs for $O(t \cdot \text{polyloglog}(N)) = O(\text{polyloglog}(N))$ times, and each Grover's iterate in each of these iterations makes $O(|W| \cdot \sqrt{N})$ queries in Line 14. Also, Line 16 uses $|W|$ many queries. Thus, the overall query cost of the algorithm is upper bounded by $O(|W| \cdot \sqrt{N} \cdot \text{polyloglog}(N))$. Since $N = 10^4 n$, we have an upper bound of $O(|W| \cdot \sqrt{n} \cdot \text{polyloglog}(n))$.

Bad event, and correctness assuming no bad event. If the estimate in Line 8 is incorrect or if the algorithm has reached Line 25 is not in W^- then we say that a bad event has occurred for Algorithm 2. We assume that these events have no happened. Thus the estimate in Line 8 is correct then $\widetilde{w}$ satisfies

$$|W^-|(1 - 1/100) \leq \widetilde{w} \leq |W^-|(1 + 1/100).$$

Define $w' = \lfloor \widetilde{w}/2 \rfloor$, thus w' satisfies the following equations.

$$\begin{aligned} |W^-|/4 \leq w' \leq |W^-|, \\ 1/2 \cdot \sqrt{|W^-|/N} \leq \sqrt{w'/N} \leq \sqrt{|W^-|/N}. \end{aligned} \tag{3}$$

Let $x = |W^-|/N$. Since $|W^-| \geq 0$ and $|W^-| \leq n$, we have

$$0 \leq x \leq 1/10^4.$$

Let $C = 1/10^4$. For $x \in [0, \sqrt{C}]$ and $A \geq 1$ (whose value is to be fixed later), define

$$g(x) = A \arcsin x/2 - \arcsin x.$$

The derivative of g is given by

$$g'(x) = \frac{A/2}{\sqrt{1 - x^2/4}} - \frac{1}{\sqrt{1 - x^2}}$$

$$\geq \frac{A}{\sqrt{4-x^2}} - \frac{1}{\sqrt{1-C}}$$
$$\geq A/2 - \frac{1}{\sqrt{1-C}}.$$

Thus for $A = 3\sqrt{1-C}$ the above derivative is positive for all $x \in [0, \sqrt{C}]$. Since $g(0) = 0$, we have, for $A \arcsin x/2 \geq \arcsin x$.

From monotonicity of arcsin in $[0, 1]$ and Equation (3) we have

$$\begin{aligned} &\arcsin(1/2 \cdot \sqrt{|W^-|/N}) \leq \arcsin(\sqrt{w'/N}) \leq \arcsin(\sqrt{|W^-|/N}) \\ &1/A \cdot \arcsin(\sqrt{|W^-|/N}) \leq \arcsin(\sqrt{w'/N}) \leq \arcsin(\sqrt{|W^-|/N}). \end{aligned} \tag{4}$$

In Line 10 we choose $\widetilde{k}$ to be $\left\lfloor \left(\frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \right\rfloor$. From Equation (4) we have

$$\begin{aligned} \left(\frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} + \frac{1}{2} \right) &\leq \left(\frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \\ &\leq (A+1) \cdot \left(\frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} + \frac{1}{2} \right). \end{aligned} \tag{5}$$

which implies

$$\begin{aligned} \left(\frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} - \frac{1}{2} \right) &\leq \left\lfloor \left(\frac{\pi}{400 \arcsin \sqrt{w'/N}} + \frac{1}{2} \right) \right\rfloor \\ &\leq (A+1) \cdot \left(\frac{\pi}{400 \arcsin \sqrt{|W^-|/N}} + \frac{1}{2} \right). \end{aligned} \tag{6}$$

From Equation (1), if we apply Grover's iterate k times then the resulting state in Line 14 is of the following form:

$$\beta \sum_{v \in W^-} |v\rangle + \sqrt{(1-\beta^2)} \sum_{v \in W^+} |v\rangle, \tag{7}$$

where $\beta = \sin((2k+1) \cdot \arcsin \sqrt{|W^-|/N})$. From Equation (6) we have

$$\frac{\pi}{200} \leq (2\widetilde{k}+1) \cdot \arcsin \sqrt{|W^-|/N} \leq (A+1)\frac{\pi}{200} + (A+2) \arcsin(\sqrt{|W^-|/N}) < \pi/2,$$

where the last inequality follows due to the choice of A ($A \leq 3$) and since $\sqrt{|W^-|/N} \leq 1/100$. Thus after $\widetilde{k}$ iterations, $\beta^2 = \sin^2((2\widetilde{k}+1) \cdot \arcsin \sqrt{|W^-|/N})$ is a constant smaller than $\pi/2$.

Since we have assumed that the bad event in Line 25 has not occurred, this means that t sample obtained is in W^-. From Equation 7 each vertex in W^- has an equal probability of being sampled. Clearly, for different iterations of the **while** loop in Line 13 the samples are independent. Also, in this case the algorithm returns in Line 21 after t iterations and hence $\Omega(t)$ uniformly distributed and independent samples from W^- are returned.

Probability of bad event. The probability of the bad event happening in Line 8 by Theorem 13 is $O(1/\text{polylog}(n))$. To upper bound the probability of the algorithm reaching Line 25, observe that with probability $\beta^2 = \Omega(1)$ (see Equation (7)) a vertex sampled in Line 15 is in the set W^-. Thus the probability that after $O(t \text{ polyloglog}(n))$, less than t vertices are seen in W^- is upper bounded by $O(1/\text{polylog}(n))$ by a Chernoff bound. ◀

Proof of Lemma 18. Consider Algorithm 3. We first analyze the query cost of the algorithm. For the correctness, we define a "bad event", argue correctness of the algorithm conditioned on the bad event not occurring, and then upper bound the probability of the bad event happening.

Query complexity. The only queries used are in Line 2 and Line 3 of the algorithm. The query cost of these steps are upper bounded by $O(|W| \cdot \sqrt{n} \cdot \text{polylog}(n))$ by Theorem 13.

Bad event, and correctness assuming no bad event. The only bad event for Algorithm 3 are that either the estimates Line 2 or Line 3 is incorrect. Let us assume that the bad event has not happened. Then

$$(1 - 1/100)|W^-| \leq \widetilde{w}_1 \leq (1 + 1/100)|W^-|,$$

and

$$(1 - 1/100)|N^+(u) \cap W^-| \leq \widetilde{w}_2 \leq (1 + 1/100)|N^+(u) \cap W^-|.$$

We have

$$\frac{99}{101} \cdot \frac{|N^+(u) \cap W^-|}{|W^-|} \leq \frac{\widetilde{w}_2}{\widetilde{w}_1} \leq \frac{101}{99} \cdot \frac{|N^+(u) \cap W^-|}{|W^-|}.$$

Thus if $|N^+(u) \cap W^-|/|W^-| \geq 1/5$ then $\widetilde{w}_2/\widetilde{w}_1 \geq 99/505$ and if $|N^+(u) \cap W^-|/|W^-| \leq 1/10$ then $\widetilde{w}_2/\widetilde{w}_1 \leq 101/990$.

Probability of bad event. By Theorem 13 and a union bound, the probability of the bad event is upper bounded by $O(1/\text{polylog}(n))$. ◀

A Generalized Quantum Branching Program

Debajyoti Bera
Department of Computer Science, IIIT-D, New Delhi, India

Tharrmashastha SAPV
Department of Computer Science, IIIT-D, New Delhi, India

Abstract

Classical branching programs are studied to understand the space complexity of computational problems. Prior to this work, Nakanishi and Ablayev had separately defined two different quantum versions of branching programs that we refer to as NQBP and AQBP. However, none of them, to our satisfaction, captures the intuitive idea of being able to query different variables in superposition in one step of a branching program traversal. Here, we propose a quantum branching program model, referred to as GQBP, with that ability. To motivate our definition, we explicitly give examples of GQBP for n-bit Deutsch-Jozsa, n-bit Parity, and 3-bit Majority with optimal lengths. We then show several equivalences, namely, between GQBP and AQBP, GQBP and NQBP, and GQBP and query complexities (using either oracle gates or a QRAM to query input bits). In a way, this unifies the different results that we have for the two earlier branching programs and also connects them to query complexity. We hope that GQBP can be used to prove space and space-time lower bounds for quantum solutions to combinatorial problems.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Quantum query complexity; Theory of computation → Quantum complexity theory

Keywords and phrases Quantum computing, quantum branching programs, quantum algorithms, query complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.31

1 Introduction

The currently popular NISQ model of quantum computing highlights the importance of quantum space, i.e., qubit usage of quantum algorithms. Motivated by the need to study the quantum space complexity of combinatorial problems, we naturally looked towards quantum branching programs. A branching program is like a "random access finite-state machine where only one input character can be looked up in any state" [4]. In the classical domain, branching programs captures the non-uniform space complexity of a problem in a very natural way [7]. A branching program of size s, i.e., with s states, can be simulated by any (non-uniform) model with $\log(s)$ space to store the state space. Similarly, if a computation involves c distinct internal states or configurations (e.g., a $\log(c)$-space-bounded Turing machine), it is possible to simulate the computation by a c-size branching program. So, e.g., polynomial-size branching programs are equivalent to the non-uniform class $LOGSPACE/poly$.

Our interest in quantum branching programs is also related to a "more practical" motivation related to a current trend in machine learning – that of explainability. A decision tree is arguably the most intuitive machine learning model that can be explained in a straightforward manner. Masek, in his Master's thesis, studied branching programs in the name of decision graphs, which are essentially compact forms of decision trees. We believe that quantum branching programs may be a worthy machine learning model that can also be explained.

We are aware of two prior attempts to define a quantum version of branching programs. They, as well as ours, follow the general idea of changing the current state (which could be a superposition of basis states) by querying some input bit; the difference primarily lies

© Debajyoti Bera and Tharrmashastha SAPV;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 31; pp. 31:1–31:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in how evolution happens. In the first version defined by Nakanishi et al. [6] (we refer to this as NQBP), the entire state is measured after each step, and evolution is continued if neither an accept nor a reject state is observed. In the version defined by Ablayev et al. [1], in each step of the evolution, a fixed bit is queried; we refer to these programs as AQBP). A closely associated model of quantum decision diagrams was also studied by Willie et al. in [9]. However, their study focuses on the design and validation of quantum circuits.

Quite a few results are known for NQBPs and AQBPs. The first advantage of quantum branching programs, more specifically NQBP, was demonstrated by Nakanishi et al. in [6]. They defined a function, namely *Half Hamming weight function*, that checks if the Hamming weight of the input is exactly $n/2$. They showed that there exists an ordered bounded width NQBP (also known as an NQOBDD) that can compute the half Hamming weight function while no ordered bounded width PBP can compute that function.

This was followed by the work of Ablayev et al [1] in which they demonstrated that an $(O(\log p_n), n)$ AQBP can compute the Mod_{p_n} function with one-sided error. However, any stable probabilistic OBDD needs width at least p_n. Here, Mod_{p_n} function is the function that decides if the Hamming weight of the input is divisible by a prime p_n. They also gave the seminal result that the class of languages in $NC1$, the languages that are decidable by uniform Boolean circuits of $O(\log n)$ depth and polynomial number of gates with at most two inputs, lies within the class of languages decidable by width-2 AQBPs [2].

Later, Sauerhoff et al. in [8] proved two results that exhibit the power and limitation of NQBP. First, they showed that the *permutation matrix test function*, the function that tests if the given matrix is a permutation matrix, can be computed using a small sized NQOBDD but needs a large deterministic OBDD. Alternatively, they also proved that any NQOBDD that computes the *disjointness function* requires $O(2^{\Omega(n)})$ size, but there exists a small sized deterministic OBDD that computes the function. More recently, in [5], Maslov et al. demonstrated that any symmetric Boolean function can be computed by $(2, O(n^2))$-AQBP. This is a striking result because there are symmetric functions that need at least 3 bits of computational space to be computed in polynomial time with arbitrary error and symmetric functions with non-trivial Fourier spectrum are not computable in polynomial time. Observe that a trivial lower bound on AQBP for computing any non-constant symmetric function is $\Omega(n)$ because the function is oblivious to the exact input and depends only on the Hamming weight of the input.

Despite the above encouraging results, we observed that AQBP suffers from the restriction of querying the same input bit within the different superposition paths in any single level. Further, the idea of measuring after every evolution step in an NQBP was not a very agreeable one. Therefore, we defined a new version which we refer to as GQBP (Figure 1 shows an example) that combines the idea of completely independent superposition paths as in NQBP with the idea of performing one measurement right at the end (as in AQBP), thus allowing interference to show its might. We use the complexity measures width (to denote the largest number of states or nodes in any level) and length (to denote the length of the longest path in the state diagram. Even though we do not consider size as a measure in this work, our results can be interpreted in terms of size by using the trivial upper bound of $size \leq width \times length$.

This paper introduces the GQBP model of branching program and relates it to a few other models like AQBP, NQBP, and query complexity. We are able to show the following.

- We show explicit general branching programs for *Parity* of n-bits, the problem of *Deutsch-Jozsa*, and *Majority* of 3 bits. These branching programs are optimal with respect to their lengths, and the optimality follows from a relation between the length of the shortest GQBP to the polynomial degree of a problem.

- We prove that an AQBP is a special form of GQBP (in which the same variable is queried in each level), and thus, the former cannot compute functions that depend non-trivially on every input bit using less than n length. We present a GQBP for Parity of length $n/2$, and this shows that GQBP does not suffer from the same limitation and is strictly more powerful, especially in the sub-linear length regime.
- Due to the above point, all the results concerning AQBP now also hold for GQBP, e.g., any problem in NC^1 can now be solved using a width-2 GQBP, and any symmetric Boolean function can be computed using a width-2, $O(n^2)$-length GQBP.
- Despite their structural difference, it is not difficult to show that a GQBP can be viewed as an NQBP. We further show that an NQBP can be simulated by a GQBP without any overhead of size and length.
- Next, we turn to quantum circuits. First, we show how to design an AQBP (and hence a GQBP) with width 2^q and length t to simulate a quantum query circuit in the QRAM model making q queries and consuming t qubits. The QRAM query model provides an alternative to implementing the oracle (in the oracle query model) by having the input in query-only registers (*aka.* QRAM) that can be queried using CNOT (or similar) gates. We want to point out that even though it may be possible to obtain a GQBP for $Parity_n$ using this approach, our handcrafted GQBP for the same uses lesser width.
- Next, we show the reverse direction, i.e., how to design a quantum circuit (using either an oracle or a QRAM to access input bits) for a given GQBP using reasonably tight depth and space (i.e., qubits). This and the above point allows us to interpret the upper and the lower bounds in quantum query complexities in terms of a GQBP and possibly derive new insights into quantum complexities of problems.

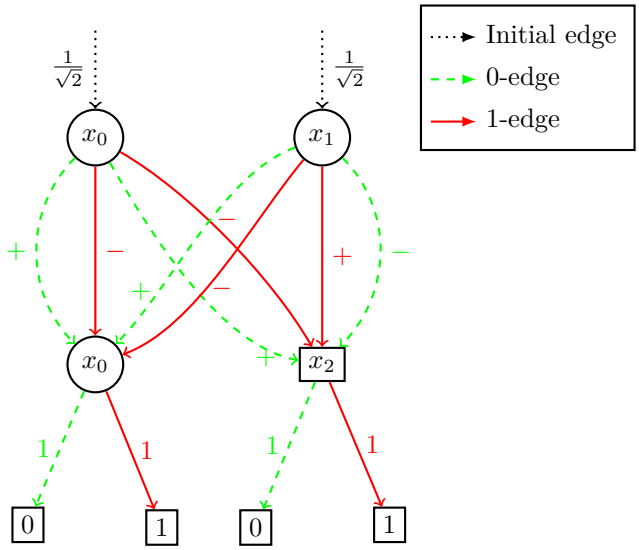


Figure 1 A length-2 and width-4 GQBP to (exactly) compute the majority of 3 bits x_0, x_1, x_2. Here, $+$ and $-$ denotes the amplitudes $\frac{1}{\sqrt{2}}$ and $-\frac{1}{\sqrt{2}}$, respectively.

2 Background: Quantum Branching Programs

In this section, we familiarize the reader with the existing branching program frameworks. We then introduce a new variant of quantum branching programs.

The most basic branching programs are the deterministic branching programs which are defined as follows.

▶ **Definition 1** (Deterministic Branching Program). *A deterministic branching program is a tuple* $(Q, E, v_0, L, \delta, F)$ *along with an evolution operator* E *where*

- Q *is a set of nodes.*
- E *is the set of edges on* Q *such that* (Q, E) *form a directed acyclic multi-graph.*
- $L : Q \to \{1, 2, \cdots, n\}$ *assigns a variable to each node.*
- $v_0 \in Q$ *is the start node.*
- $\delta : Q \otimes \{0, 1\} \to Q$ *is transition function.*
- $F \subseteq Q$ *is the set of sinks such that* $F = F_a \cup F_r$ *where* F_a *and* F_r *are the accept and reject sinks respectively.*

Given an input x, starting from the node v_0, one iteration of the evolution of the deterministic branching program consists of two steps:

1. Let the current node be v. Query the value of $x_{L(v)}$ and call it j. Move to the node $\delta(v, j) = v'$.
2. If $v' \in F_a$ then accept the input, else if $v' \in F_r$ then reject the input. Else, continue.

The complexity of a branching program is primarily categorized by two measures: *size* and *length* of a branching program.

▶ **Definition 2** (Size and length of a branching program).

- *The* ***size*** *of a branching program is the total number of nodes in the branching program.*
- *The* ***length*** *of a branching program is the length of the longest path from the start node to any sink node.*

It is well known that any classical branching program can be transformed into an equivalent levelled branching program, i.e., a branching program where all the transitions are such that they go from a node at a level i only to a node at level $i + 1$ for each i. From now on, we assume that the branching program is levelled unless otherwise stated.

For levelled branching program, another well-used measure of complexity is the *width* of a branching program.

▶ **Definition 3** (width of a leveled branching program). *The* ***width*** *of a levelled branching program is the maximum number of nodes in any level of the branching program.*

A straightforward generalization of the deterministic branching programs is the probabilistic branching programs. But before we define the probabilistic branching programs, we establish well-behaved transition functions. A transition function $\delta : Q \otimes \{0, 1\} \times Q \to \mathbb{R}$ is said to be **classically well-behaved** if for any $v \in Q$ and $a \in \{0, 1\}$,

$$\sum_{v' \in Q} \delta(v, a, v') = 1.$$

Now, the probabilistic branching programs are defined as below.

▶ **Definition 4** (Probabilistic Branching Program). *A probabilistic branching program is a tuple* $(Q, E, v_0, L, \delta, F)$ *along with an evolution operator* E *where*

- Q *is a set of nodes.*
- E *is the set of edges on* Q *such that* (Q, E) *form a directed acyclic multi-graph.*
- $L : Q \to \{1, 2, \cdots, n\}$ *assigns a variable to each node.*
- $v_0 \in Q$ *is the start node.*

- $\delta : Q \otimes \{0,1\} \times Q \to \mathbb{R}$ *is a classically well-behaved transition function.*
- $F \subseteq Q$ *is the set of sinks such that* $F = F_a \cup F_r$ *where* F_a *and* F_r *are the accept and reject sinks respectively.*

Given an input x, starting from the node v_0, one iteration of the evolution of the deterministic branching program consists of two steps:

1. Let the current node be v. Query the value of $x_{L(v)}$ and call it j. Move to the node v' with probability $\delta(v, j, v')$.
2. If $v' \in F_a$ then accept the input, else if $v' \in F_r$ then reject the input. Else, continue.

An input string is *accepted* by a branching program if the evolution ends in an accept state. In this paper, we are concerned with *exact computation*, i.e., the probability of acceptance should be 1 for good inputs and 0 for bad inputs.

In general, branching programs are complex structures to analyse. For ease of analysis, various variants of the branching programs were introduced. Some of the variants are as follows.

- **Oblivious branching programs:** These are levelled branching programs in which the variable to be queried depends on the level of the current node rather than the node itself.
- **Ordered binary decision diagrams (OBDD):** These are branching programs with a fixed ordering of querying the indices along any path.
- **Read-k branching programs:** These are branching programs in which the number of times any variable is queried is at most k along any path from the source node to a sink node.

The notion of quantum branching programs was first introduced by Nakanishi et al. in [6] as an extension of the classical probabilistic branching programs.

▶ **Definition 5** (Nakanishi's Quantum Branching Program (NQBP)). *A quantum branching program is a tuple* $(Q, E, v_0, L, \delta, O)$ *where*

- $Q = Q_{acc} \cup Q_{rej} \cup Q_{non}$ *is a set of nodes.*
- E *is the set of edges on* Q *such that* (Q, E) *is a directed multi-graph.*
- v_0 *is the start node.*
- $L : Q \to \{1, 2, \cdots, n\}$ *assigns a variable to each node.*
- $\delta : Q \otimes \{0,1\} \otimes Q \to \mathbb{C}$ *is a quantumly well-behaved transition function.*
- O *is the projective measurement induced by the set* $\left\{\Pi_{acc}, \Pi_{rej}, \Pi_{non}\right\}$ *where* Π_i *is the projection operator onto the space* $V_i = span\{|v\rangle : v \in Q_i\}$ *for each* $Q_i \in \{Q_{acc}, Q_{rej}, Q_{non}\}$.

For any fixed input x, let O_x denote an oracle defined as $O_x(i) = x_i$. Given an oracle O_x to access the input, the ith iteration of the evolution of NQBP comprises of the following steps:

1. Define an operator $U^O : Q \to Q$ as

$$U^{O_x} |v\rangle = \sum_{v' \in Q} \delta(v, O_x(L(v)), v') |v'\rangle .$$

2. Apply U^{O_x} on $|\psi_i\rangle = U^{O_x} |\psi_{i-1}\rangle$, $|\psi_0\rangle = |v_0\rangle$.
3. Observe $|\psi_i\rangle$ with respect to O.
4. If the post-measurement state is in the subspace spanned by $|E_{non}\rangle$, continue. Else accept or reject depending on the outcome.

A transition function $\delta : Q \otimes \{0,1\} \otimes Q \to \mathbb{C}$ is said to be **quantumly well-behaved** if for any $v_1, v_2 \in Q$ and all inputs $x \in \{0,1\}^n$,

$$\sum_{v' \in Q} \delta(v_1, x_{L(v_1)}, v')\delta^*(v_2, x_{L(v_2)}, v') = \begin{cases} 1 & \text{if } v_1 = v_2 \\ 0 & \text{if } v_1 \neq v_2 \end{cases}$$

Closely followed by this work, Ablayev et al., in [1], proposed a new model for quantum branching programs that was significantly different from NQBPs. Before we define Ablayev's model, we first define a quantum transformation as defined by Ablayev et al.

▶ **Definition 6** (Quantum Transformation). *Given an input string $x \in \{0,1\}^n$, a d-dimensional quantum transformation $\langle j, U(0), U(1)\rangle$ on a state $|\psi\rangle \in \mathcal{H}^{\otimes d}$ is defined as*

$$|\psi'\rangle = U(x_j)\,|\psi\rangle\,.$$

where $U(0)$ and $U(1)$ are a $d \times d$ unitary matrices. Intuitively, the quantum transformation applies $U(0)$ to $|\psi\rangle$ if the jth bit of x is 0 and applies $U(1)$ if the jth bit of x is 1.

Now, we define Ablayev's quantum branching program model.

▶ **Definition 7** (Ablayev's Quantum Branching Programs (AQBP)). *A quantum branching program P of width d and length l is a tuple $\langle T, |\psi_0\rangle, F\rangle$ where*

- *$T = \left(\langle j_i, U_i(0), U_i(1)\rangle\right)_{i=1}^{l}$ is a sequence of d-dimensional quantum transformations.*
- *$|\psi_0\rangle$ is the initial configuration of P.*
- *$F \subseteq \{|0\rangle, |1\rangle, \cdots, |d-1\rangle\}$ is the set of accepting states.*

Given an input $x = x_1 x_2 \cdots x_n$, the AQBP P transforms as

$$|\psi_x\rangle = U_l(x_{j_l})U_{l-1}(x_{j_{l-1}}) \cdots U_2(x_{j_2})U_1(x_{j_1})\,|\Psi_0\rangle\,.$$

To obtain the output, the state $|\psi_x\rangle$ is measured with projection matrix M such that $M_{ii} = 1$ iff $i \in F$ and 0 else. The probability that P accepts the input x is

$$P_{accept}(x) = ||M\,|\psi_x\rangle\,||^2$$

Notice that this model is the quantum equivalent of the oblivious classical branching programs; the query made at any given level is independent of the nodes in that level of the AQBP.

We define a new model of branching programs that differs from the above two models.

▶ **Definition 8** (Generalized quantum branching program (GQBP)). *A quantum branching program is a tuple $(Q, E, |v_0\rangle, L, \delta, F)$ where*

- *$Q = \bigcup_{i=0}^{l} Q_i$ is a set of nodes where Q_i is the set of nodes at level i.*
- *E is a set of edges between the nodes in Q such that (Q, E) is a levelled directed acyclic multi-graph.*
- *$|v_0\rangle$ is the initial state which is a superposition of the nodes in Q_0.*
- *$\delta : Q \times \{0,1\} \times Q \to \mathbb{C}$ is a **quantumly well-behaved** transition function.*
- *$L : Q \to \{1, 2 \cdots, n\}$ is a function that assigns a variable to each node in $Q \setminus F$.*
- *$F \subseteq Q$ is the set of terminal nodes.*

The evolution of this GQBP is defined as follows:

1. Given access to an input x using oracle O_x, define $U_i^{O_x} : \mathcal{H}(d) \to \mathcal{H}(d)$ for $i \in \{1, 2, \cdots, l\}$ that has query access to O_x as

$$U_i^{O_x} |v\rangle = \sum_{v' \in Q_i} \delta(v, O_x(L(v)), v') |v'\rangle$$

for any $|v\rangle \in Q_{i-1}$ where $\mathcal{H}(d)$ is the Hilbert space on d dimensions with d as the width of the QBP.
2. Start at $|v_0\rangle$.
3. For $i = 1$ to l, perform

$$|\psi_i\rangle = U_i^{O_x} |\psi_{i-1}\rangle$$

where $|\psi_0\rangle = |v_0\rangle$.
4. Measure $|\psi_l\rangle$ in the standard basis as m. If $|m\rangle \in F$ then accept the input. Else, reject the input.

We say that a QBP $\mathcal{G}$ **exactly computes** a function f if for all inputs x, $\mathcal{G}$ accepts x if $f(x) = 1$ and rejects x if $f(x) = 0$, both with probability 1. Similarly, a QBP $\mathcal{G}$ is said to **approximate** a f if for all inputs x, $\mathcal{G}$ accepts x if $f(x) = 1$ and rejects x if $f(x) = 0$, both with probability at least $2/3$.

It is quite straightforward that this model of quantum branching program subsumes the AQBP model. To see this in detail, we direct the readers to the proof of Theorem 14. So, the set of languages computable by AQBPs is a subset of that computable by GQBPs.

3 Upper Bounds on GQBP

In this section, we show some GQBPs for some simple Boolean functions. We first start with the classic Deutsch-Josza problem, which is a promise problem whose task is to identify if an n-bit binary string is balanced (has Hamming weight $n/2$) or is constant (has Hamming weight 0 or n). We define the notation $\texttt{GQBP}_{w,l}$ to denote the class of all problems that are computable by some w-width l-length GQBP exactly.

▶ **Theorem 9.** $DJ_n \in \mathit{GQBP}_{n,1}$.

Proof. Consider the two-layered GQBP $\mathcal{G} = (Q, E, |v_0\rangle, L, \delta, F)$ where $Q = Q_0 \cap Q_1$, $Q_i = \{v_{i,j} : j \in \{1, \cdots, n\}\}$, $E = \{(v_{0,i}, v_{1,j}) : i, j \in \{1, \cdots, n\}\}$ and $F = Q_1 \setminus \{|v_{1,1}\rangle\}$. Define the function L as $L(v_{0,i}) = x_i$ for $i \in \{1, \cdots, n\}$. Also, define the transition function δ as

$$\delta(v_{0,i}, a, v_{1,j}) = \frac{1}{2^{n/2}} (-1)^a (-1)^{i \cdot j}$$

for all $i, j \in \{1, \cdots, n\}$. Now, set $|v_0\rangle = \frac{1}{2^{n/2}} \sum_{v_{0,i} \in Q_0} |v_{0,i}\rangle$. Fix an input x. Then the transition unitary $U_1^{O_x}$ will be defined as

$$U_0^{O_x} |v_{0,i}\rangle = \sum_{v_{1,j} \in Q_1} \delta(v_{0,i}, x_i, v_{1,j}) |v_{1,j}\rangle = \frac{1}{2^{n/2}} \sum_{v_{1,j} \in Q_1} (-1)^{x_i} (-1)^{i \cdot j} |v_{1,j}\rangle .$$

Then the final state of $\mathcal{G}$ before measurement can be calculated as

$$\begin{aligned} U_0^{O_x} |v_0\rangle &= \frac{1}{2^n} \sum_{v_{0,i} \in Q_0} U_0^{O_x} |v_{0,i}\rangle \\ &= \frac{1}{2^n} \sum_{v_{0,i} \in Q_0} \frac{1}{2^{n/2}} \sum_{v_{1,j} \in Q_1} (-1)^{x_i} (-1)^{i \cdot j} |v_{1,j}\rangle \\ &= \frac{1}{2^n} \sum_{v_{1,j} \in Q_1} \Big[\sum_{v_{0,i} \in Q_0} (-1)^{x_i \oplus i \cdot j} \Big] |v_{1,j}\rangle \end{aligned}$$

Since, $F = Q_1 \setminus \{|v_{1,1}\rangle\}$, the probability of accepting the input x is

$$Pr_{\mathcal{G}}[accept] = 1 - \left| \sum_{v_{0,i} \in Q_0} (-1)^{x_i} \right|^2 = \begin{cases} 1, & \text{if } x \text{ is balanced} \\ 0, & \text{if } x \text{ is constant} \end{cases}$$ ◀

▶ **Theorem 10.** $Parity_n \in \mathtt{GQBP}_{2,n/2}$.

Proof. Observe that any classical branching program that solves the problem with probability 1 must query all the bits in some path and so must have length n. To reduce the length, our GQBP uses superposition to query each half separately, "compute" their parity in each branch, and then use a simple decision graph to output the xor of those parities. The GQBP is illustrated in Figure 2. [1]

Without loss of generality, we assume that n is even. To construct a GQBP for $Parity_n$, set $Q = \big\{v_{i,a} : i \in \{0, 1, \cdots, n/2\}, a \in \{0, 1\}\big\}$, $|v_0\rangle = \frac{1}{\sqrt{2}}(|v_{0,0}\rangle + |v_{0,1}\rangle)$, $E = \big\{(v_{i,a}, v_{i+1,b}) : i \in \{0, 1, \cdots, n/2\}, a \in \{0, 1\}\big\}$ and $F = \{v_{n/2,1}\}$. Let $l = n/2$. As for the transition function δ, for the first $l - 1$ layers, i.e., $i \in \{0, 1, \cdots, l - 2\}$ define

$$\delta(v_{i,j}, a, v_{p,q}) = \begin{cases} 1, \text{ if } p = i + 1, j = q, a = 0 \\ -1, \text{ if } p = i + 1, j = q, a = 1 \\ 0, \text{ else} \end{cases}$$

This gives that $U_i^{O_x} |v_{i-1,a}\rangle = (-1)^{L(v_{i-1,a})} |v_{i,a}\rangle$ for all $a \in \{0, 1\}$. For the nodes $v_{l-1,0}$ and $v_{l-1,1}$, define

$$\delta(v_{l-1,0}, 0, v_{l,0}) = \delta(v_{l-1,0}, 0, v_{l,1}) = \delta(v_{l-1,1}, 0, v_{l,0}) = \delta(v_{l-1,1}, 1, v_{l,1}) = 1/\sqrt{2}$$

and

$$\delta(v_{l-1,0}, 1, v_{l,0}) = \delta(v_{l-1,0}, 1, v_{l,1}) = \delta(v_{l-1,1}, 1, v_{l,0}) = \delta(v_{l-1,1}, 0, v_{l,1}) = -1/\sqrt{2}.$$

So, we have

$$U_{l-1}^{O_x} |v_{l-1,0}\rangle = \frac{(-1)^{L(v_{l-1,0})}}{\sqrt{2}} \Big(|v_{l,0}\rangle + |v_{l,1}\rangle \Big)$$

and

$$U_{l-1}^{O_x} |v_{l-1,1}\rangle = \frac{(-1)^{L(v_{l-1,1})}}{\sqrt{2}} \Big(|v_{l,0}\rangle - |v_{l,1}\rangle \Big).$$

[1] It is tempting to try this idea to reduce the length even further, e.g., to $n/4$ by creating a superposition of 4 branches. That approach would get stuck at the last stage of combining all the partial parities. In fact, we show later that it is not possible to reduce the length beyond $n/2$ (13).

Next, define the function L as $L(v_{i,j}) = 2 * i + j$. Then, starting from the state $|v_0\rangle$, the state evolves through the first $l-1$ layers as follows,

$$
\begin{aligned}
|v_0\rangle &= \frac{1}{\sqrt{2}}(|v_{0,0}\rangle + |v_{0,1}\rangle) \\
&\xrightarrow{U_1^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0}|v_{1,0}\rangle + (-1)^{x_1}|v_{1,1}\rangle) \\
&\xrightarrow{U_2^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0 \oplus x_2}|v_{2,0}\rangle + (-1)^{x_1 \oplus x_3}|v_{2,1}\rangle) \\
&\cdots \\
&\xrightarrow{U_i^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0 \oplus x_2 \oplus \cdots \oplus x_{2i-2}}|v_{i,0}\rangle + (-1)^{x_1 \oplus x_3 \oplus \cdots \oplus x_{2i-1}}|v_{i,1}\rangle) \\
&\cdots \\
&\xrightarrow{U_{l-1}^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0 \oplus x_2 \oplus \cdots \oplus x_{n-4}}|v_{l-1,0}\rangle + (-1)^{x_1 \oplus x_3 \oplus \cdots \oplus x_{n-3}}|v_{l-1,1}\rangle) = |\psi_{l-1}\rangle \text{ (say)}
\end{aligned}
$$

For the last transition, the state evolves as

$$
\begin{aligned}
|\psi_{l-1}\rangle &= \frac{1}{\sqrt{2}}((-1)^{x_0 \oplus x_2 \oplus \cdots \oplus x_{n-4}}|v_{l-1,0}\rangle + (-1)^{x_1 \oplus x_3 \oplus \cdots \oplus x_{n-3}}|v_{l-1,1}\rangle) \\
&\xrightarrow{U_l^{O_x}} \frac{1}{\sqrt{2}}\Bigg[(-1)^{x_0 \oplus x_2 \oplus \cdots \oplus x_{n-4}}\Big[\frac{(-1)^{x_{n-2}}}{\sqrt{2}}\Big(|v_{l,0} + |v_{l,1}\rangle\rangle\Big)\Big] \\
&\qquad + (-1)^{x_1 \oplus x_3 \oplus \cdots \oplus x_{n-3}}\Big[\frac{(-1)^{x_{n-1}}}{\sqrt{2}}\Big(|v_{l,0}\rangle - |v_{l,1}\rangle\Big)\Big]\Bigg] \\
&= \frac{1}{2}\Bigg[\Big((-1)^{x_0 \oplus x_2 \oplus \cdots \oplus x_{n-2}} + (-1)^{x_1 \oplus x_3 \oplus \cdots \oplus x_{n-1}}\Big)|v_{l,0}\rangle \\
&\qquad + \Big((-1)^{x_0 \oplus x_2 \oplus \cdots \oplus x_{n-2}} - (-1)^{x_1 \oplus x_3 \oplus \cdots \oplus x_{n-1}}\Big)|v_{l,1}\rangle\Bigg]
\end{aligned}
$$

So, if $x_0 \oplus x_2 \oplus \cdots x_{n-2} = x_1 \oplus x_3 \oplus \cdots x_{n-1}$ or alternatively, $Parity_n(x) = 0$, then $|\psi_l\rangle = |v_{l,0}\rangle$ and the input will be rejected with probability 1 post measurement. Else if $x_0 \oplus x_2 \oplus \cdots x_{n-2} \neq x_1 \oplus x_3 \oplus \cdots x_{n-1}$, i.e, if $Parity_n(x) = 1$, then $|\psi_l\rangle = |v_{l,1}\rangle$. In this case, the input will be accepted with probability 1 post measurement. ◀

It is noteworthy that any query circuit requires at least $\log(n)$ qubits to query the oracle for an index. Then as a derivative of Corollary 21, one would be led to think that any GQBP would require width at least n. However, in contradiction to that, Theorem 10 gives us a width-2 GQBP that computes the $Parity_n$ function exactly. This is a classic example that shows that space lower bounds on query circuits does not hold for the GQBPs. On the contrary, later in section 5, we will show that asymptotic query lower bounds of query circuits also hold for the GQBPs and that of GQBPs hold for query circuits.

▶ **Lemma 11.** $Maj_3 \in \mathit{GQBP}_{3,2}$

Proof. The idea behind the construction is first to compare the first two bits. If they are the same, their value is the majority. Otherwise, the value of the third bit turns out as the majority. The state diagram is illustrated in Figure 1.

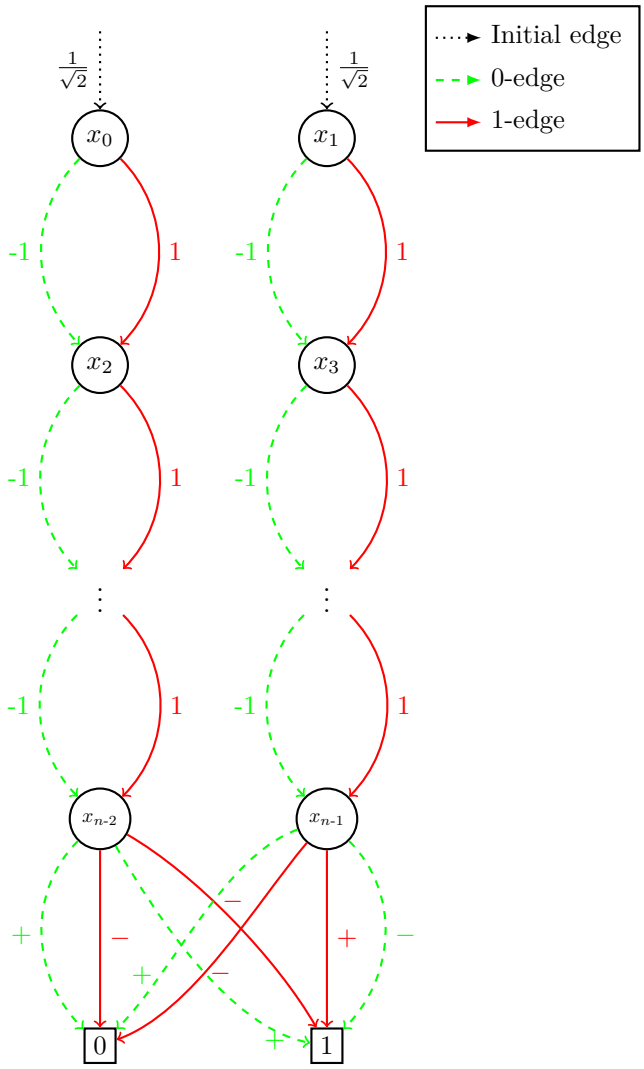


Figure 2 A GQBP for the n-bit Parity problem. $+$ and $-$ indicates transitions with amplitudes $\frac{1}{\sqrt{2}}$ and $-\frac{1}{\sqrt{2}}$, respectively.

Consider a 3-levelled GQBP $\mathcal{G} = (Q, E, |v_0\rangle, L, \delta, F)$ for Maj_3 where the set of nodes $Q = \{v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}, v_{2,0}, v_{2,1}, v_{2,2}, v_{2,3}\}$, $|v_0\rangle = \frac{1}{\sqrt{2}}(v_{0,0} + v_{0,1})$, the function L is such that $L(v_{0,0}) = L(v_{1,0}) = x_1, L(v_{0,1}) = x_2$ and $L(v_{1,1} = x_3)$, and $F = \{|v_{2,1}\rangle, |v_{2,3}\rangle\}$. Define the transition function δ as

$$\delta(v_{0,0}, 0, v_{1,0}) = \delta(v_{0,0}, 0, v_{1,1}) = \delta(v_{0,1}, 0, v_{1,0}) = \delta(v_{0,1}, 1, v_{1,1}) = 1/\sqrt{2}$$
$$\delta(v_{0,0}, 1, v_{1,0}) = \delta(v_{0,0}, 1, v_{1,1}) = \delta(v_{0,1}, 1, v_{1,0}) = \delta(v_{0,1}, 0, v_{1,1}) = -1/\sqrt{2}$$

and

$$\delta(v_{1,0}, 0, v_{2,0}) = \delta(v_{1,0}, 1, v_{2,1}) = \delta(v_{1,1}, 0, v_{2,2}) = \delta(v_{1,1}, 0, v_{2,3}) = 1.$$

From this transition function, we obtain the transition unitary $U_1^{O_x}$ as

$$U_1^{O_x} |v_{0,0}\rangle = \frac{(-1)^{L(v_{0,0})}}{\sqrt{2}} \Big(|v_{1,0}\rangle + |v_{1,1}\rangle \Big) = \frac{(-1)^{x_0}}{\sqrt{2}} \Big(|v_{1,0}\rangle + |v_{1,1}\rangle \Big)$$

and

$$U_1^{O_x}|v_{0,1}\rangle = \frac{(-1)^{L(v_{0,1})}}{\sqrt{2}}\Big(|v_{1,0}\rangle - |v_{1,1}\rangle\Big) = \frac{(-1)^{x_1}}{\sqrt{2}}\Big(|v_{1,0}\rangle - |v_{1,1}\rangle\Big).$$

Now, one step of the evolution of the program can be given as

$$\begin{aligned}
|v_0\rangle &= \frac{1}{\sqrt{2}}(|v_{0,0}\rangle + |v_{0,1}\rangle) \\
&\xrightarrow{U_1^{O_x}} \frac{1}{\sqrt{2}}\left(\frac{(-1)^{x_0}}{\sqrt{2}}\Big(|v_{1,0} + |v_{1,1}\rangle\rangle\Big) + \frac{(-1)^{x_1)}}{\sqrt{2}}\Big(|v_{1,0}\rangle - |v_{1,1}\rangle\Big)\right) \\
&= \frac{1}{2}\left(\Big((-1)^{x_0} + (-1)^{x_1}\Big)|v_{1,0}\rangle + \Big((-1)^{x_0} - (-1)^{x_1}\Big)|v_{1,1}\rangle\right) \\
&= \begin{cases} |v_{1,0}\rangle, & \text{if } x_0 = x_1 \\ |v_{1,1}\rangle, & \text{if } x_0 \neq x_1 \end{cases} \\
&\xrightarrow{U_2^{O_x}} \begin{cases} |v_{2,0}\rangle, & \text{if } x_0 = x_1 \text{ and } x_0 = 0 \\ |v_{2,1}\rangle, & \text{if } x_0 = x_1 \text{ and } x_0 = 1 \\ |v_{2,2}\rangle, & \text{if } x_0 \neq x_1 \text{ and } x_2 = 0 \\ |v_{2,3}\rangle, & \text{if } x_0 \neq x_1 \text{ and } x_2 = 1 \end{cases}
\end{aligned}$$

We have $F = \{|v_{2,1}\rangle, |v_{2,3}\rangle\}$. This implies that if the given input is such that $x_0 = x_1$ and $x_0 = 1$ or $x_0 \neq x_1$ and $x_2 = 1$, then the GQBP accepts the input. Now, see that if $x_0 = x_1$, clearly, the value of x_0 is the majority. However, if $x_0 \neq x_1$, then the value of x_2 is the majority. Putting them together, we have that the GQBP $\mathcal{G}$ computes the Maj_3 function exactly. ◀

On careful observation, one can notice that the amplitudes of the basis states of the state of a GQBP after the evolution is a polynomial of the input x since, at each node, the outgoing edges are decided based on the value of the query made at that node. Moreover, each step of the evolution (applying one $U_i^{O_x}$) increases the degree of the polynomial corresponding to the amplitudes by at most 1. So, clearly, the amplitude of any node of the final state of an l-length GQBP can be given by a polynomial of degree of at most l. For any GQBP, the probability of accepting a given input is the sum of the probabilities of obtaining the nodes in F on measuring the final state. Consequently, we have the probability that an l-length GQBP accepts an input x is a polynomial of degree at most $2l$. This observation leads us to the following theorem.

▶ **Theorem 12.** *Let f be a Boolean function. Let $deg(f)$ and $\widetilde{deg}(f)$ be the exact degree and approximate degree of f. Then,*

1. *Any GQBP that exactly computes f should have length at least $deg(f)/2$.*
2. *Any GQBP that approximates f should have length at least $\widetilde{deg}(f)/2$.*

As a result, we obtain the following lower bound that shows that our constructions above are tight with respect to length.

▶ **Theorem 13.** *$Maj_3 \notin \texttt{GQBP}_{w,1}$ and $Parity_n \notin \texttt{GQBP}_{w,t}$ for any $w \in \mathbb{N}$ and any $t < \frac{n}{2}$.*

4 Relation Between GQBP and other variants

Similar to $\texttt{GQBP}_{w,l}$, we define the complexity class $\texttt{AQBP}_{w,l}$ as the class of problems that are computed exactly by some w-width l-length AQBP. Then, we have the following result.

▶ **Theorem 14.** $AQBP_{w,l} \subsetneq GQBP_{w,l}$

Proof. First, we show that any (w,l)-AQBP is also a (w,l)-GQBP. Let $\mathcal{A} = \langle T, |\psi_0\rangle, F_a\rangle$ be a (w,l)-AQBP where $T = \Big(\langle j_i, U_i(0), U_i(1)\rangle\Big)_{i=1}^{l}$. Construct a (w,l)-GQBP $\mathcal{G}$ $=(Q, E, v_0, L, \delta, F_g)$ as follows. Set $Q = \{v_{(s,t)} : s \in \{0, 1, \cdots, l\}, t \in \{0, 1, \cdots, d-1\}\}$ and $F_g = \{v_{(l,t)} : |t\rangle \in F_a\}$. Let $v_0 = v_{(0,0)}$. Now, define a function $L : Q \setminus F \longrightarrow \{1, 2 \cdots, n\}$ as $L(v_{(s,t)}) = j_s$ for all t. Define the transition function $\delta : Q \times \{0,1\} \times Q \longrightarrow \mathbb{C}$ as

$$\delta(v_{s,t}, a, v_{p,q}) = \begin{cases} 0 \text{ if } p \neq s+1 \\ U_s(a)[q,t] \text{ if } p = s+1 \end{cases}$$

It is easy to see that δ is quantumly well-behaved.

To show that $\mathcal{G}$ simulates $\mathcal{A}$ exactly, it suffices to show that that the unitary $U_i^{O_x} = U_i(j_i)$ for all $i \in [l]$ since $|v_{0,0}\rangle = |v_0\rangle$ and $F_g = F$. Fix an input x. Let $i \in [l]$. Then from definition of GQBP, we have that for any $v \in Q_i$

$$U_i^{O_x} |v\rangle = \sum_{v' \in Q_{i+1}} \delta(v, L(v), w) |v'\rangle$$

We know that $L(v) = j_i$ for any $v \in Q_i$. So, we have

$$U_i^{O_x} = \sum_{v \in Q_i} \left[\sum_{v' \in Q_{i+1}} \delta(v, L(v), v') |v'\rangle \right] \langle v| = \sum_{v \in Q_i} \sum_{v' \in Q_{i+1}} U_i(j_i)[v', v] |v'\rangle \langle v| = U_i(j_i).$$

Hence, $\mathcal{G}$ simulates $\mathcal{A}$ exactly. This shows that any arbitrary (w,l)-AQBP can be simulated by an equivalent (w,l)-GQBP.

To show the tight inclusion, observe that an AQBP requires length at least n to query all the bits of an n-length input. Thus, $Parity_n \notin \texttt{AQBP}_{w,t}$ for any w and any $t < n$; however, we showed in the earlier section that $Parity_n \in \texttt{GQBP}_{2,n/2}$. ◀

Next, we turn to NQBP. Observe that a GQBP can be written as an NQBP if we set $Q_{acc} = F$, Q_{rej} as the states not in F in the last layer of the GQBP, and Q_{non} as the rest of the states. Since the evolution of a GQBP happens in layers, in none of the evolutions but the last one will any state in $|E_{acc}\rangle$ and $|E_{rej}\rangle$ be observed, and hence the evolution would continue uninterrupted. And since there is no non-E_{non} node in the last layer, the evolution would stop after that. Thus, a size-s length-l GQBP is also a size-s length-l NQBP.

Next, we prove the reverse direction.

▶ **Theorem 15.** *Any size s length l NQBP can be simulated by a width s length l GQBP.*

Proof. Let $\mathcal{N}$ be an NQBP of size s and length l. Now, make l many copies of $\mathcal{N}$. For any edge between two nodes v, v' in $\mathcal{N}$, remove that edge and add an edge between node v of copy i and node v' of copy $i+1$ for all $i \in [l-1]$. Clearly, this program is levelled since at i^{th} step, the state of the program is in some superposition of the nodes of copy i. Since, the size of $\mathcal{N}$ is s, the space of the superposition is $\mathcal{H}(s)$. ◀

5 Relation Between GQBPs and Quantum Query Circuits

To the best of our knowledge, there does not exist any study of the relation between quantum circuits and quantum branching programs. This section presents complete reductions from the GQBPs to the quantum query circuits and vice-versa. As an add-on, we also present reductions from the AQBPs to the quantum circuits with QRAM and standard oracle access to inputs.

We define a quantum circuit with QRAM[2] input access as a quantum circuit with two registers, a read-only input register and a workspace register, and a series of controlled-unitary gates with the control on exactly one of the qubit of the input register and the target gate on the workspace register. For any QRAM circuit C, the number of qubits in the workspace register is the space or qubit complexity of C, and the number of controlled gates is the query complexity of C. Mathematically, a q-qubit t-query quantum circuit with QRAM access to inputs can be described as $C = \tilde{O}^{p_t} \cdots \tilde{O}^{p_1}$, where the controlled-gate $\tilde{O}^{p_k}$ is defined as

$$\tilde{O}^{p_k} = \left(\mathbb{I}^{p_k-1} \otimes |0\rangle\langle 0| \otimes \mathbb{I}^{n-p_k}\right) \otimes \tilde{U}_k^{(0)} + \left(\mathbb{I}^{p_k-1} \otimes |1\rangle\langle 1| \otimes \mathbb{I}^{n-p_k}\right) \otimes \tilde{U}_k^{(1)}$$

where $\tilde{U}_k^{(0)}$ and $\tilde{U}_k^{(1)}$ are q-qubit unitaries.

▶ **Theorem 16.** *For any q-qubit, t-query quantum circuit C with QRAM query model, there exists an $(2^q, t)$-AQBP that simulates C.*

Proof. Let C be a q-qubit t-query quantum circuit with a QRAM query model. Let C be given as $C = \tilde{O}^{p_t} \cdots \tilde{O}^{p_1}$. Without loss of generality, let the initial state be $|0\rangle$. Let F_c be the set of basis states which, on obtaining post-measurement the given input is accepted.

Construct an AQBP $\mathcal{A} = \langle T, |\psi_0\rangle, F_A\rangle$ as follows. Set $|\psi_0\rangle = |0\rangle$ and $F_A = F_c$. Define $T = (\langle j_i, U_i(0), U_i(1)\rangle)_{i=1}^t$ where for any $i \in \{1, \cdots, t\}$, we set $j_i = p_i$, $U_i(0) = \tilde{U}_i^{(0)}$ and $U_i(1) = \tilde{U}_i^{(1)}$. We can see that $U_i(0), U_i(1)$ unitaries act on the Hilbert space $\mathcal{H}^{2^q}$. Clearly, the final state of $\mathcal{A}$ before the measurement can be obtained as

$$|\psi_f\rangle = U_t(j_t) \cdots U_1(j_1) |\psi_0\rangle = \tilde{U}_t^{(j_t)} \cdots \tilde{U}_1^{(j_1)} |0\rangle = C |0\rangle .$$

Since we have $F_A = F_c$, the probability of accepting an input x by $\mathcal{A}$ is

$$Pr_{\mathcal{A}}[accept|x] = \sum_{|i\rangle \in F_A} \left| \langle i|\psi_f\rangle \right|^2 = \sum_{|i\rangle \in F_c} \left| \langle i| C |0\rangle \right|^2 = Pr_C[accept|x].$$

This gives that the $(2^q, t)$-AQBP $\mathcal{A}$ exactly simulates the circuit C. ◀

▶ **Corollary 17.** *For any q-qubit, t-query quantum circuit C with QRAM query model, there exists an $(2^q, t)$-GQBP that simulates C.*

We show the following reductions from an AQBP to a quantum circuit.

▶ **Theorem 18.**

1. *Any (d, l)-AQBP can be simulated using a circuit with $\log d$ qubits and l gates that have a control on the input in the QRAM model when the input is stored in the QRAM.*
2. *Any (d, l)-AQBP can be simulated using a circuit with $\log n + \log d + 1$ qubits and $2 \cdot l$ queries to the oracle when the input is accessible only using an oracle.*

[2] QRAM stands for Quantum Random Access Memory.

The proof of Theorem 18 is presented in Appendix A. Although the reduction from an AQBP to a query circuit looks trivial, the same is not true for a reduction from a GQBP to a query circuit. Moreover, a generic reduction from a GQBP to a query circuit under the QRAM model is impossible. This is because of the fact that the QRAM model, as defined previously, does not allow for superposition queries, while GQBPs can query in superposition.

We now show that any quantum query circuit can be simulated by a GQBP exactly.

▶ **Theorem 19.** *For any q-qubit t-query quantum circuit C that has access to the input through a standard oracle, there exists an $\left(2^q, 2t+1\right)$-GQBP that simulates C.*

Proof. On an outline, to simulate C, we construct a GQBP such that all the odd transitions correspond to the unitaries and the even transitions correspond to the oracle calls. For this proof we assume that the query oracle used is a phase oracle, i.e., the oracle O_x acts as $O_x \left|i\right\rangle = (-1)^{x_i} \left|i\right\rangle$. Let the circuit C be given as $C = \tilde{U}_t O_x \tilde{U}_{l-1} \cdots \tilde{U}_1 O_x \tilde{U}_0$ and let F_c be the set of basis states which on obtaining post measurement the given input is accepted. Fix $l = 2t+1$ and $w = 2^q$. Now, define a GQBP $\mathcal{G} = (Q, E, v_0, L, \delta, F)$. Set $Q = \{v_{i,j} : i \in \{0, 1, \cdots, l\}, j \in \{0, 1, \cdots, w-1\}\}$, $E = \{(v_{i,j}, v_{i+1,k}) : i \in \{0, 1, \cdots, l\}$ and $j, k \in \{0, 1, \cdots, w-1\}\}$ and $\left|v_0\right\rangle = \left|v_{0,0}\right\rangle$ Also, fix $F = \{v_{l,j} : \left|j\right\rangle \in F_c\}$. Define the transition function δ as follows:

$$\delta(v_{i,j}, a, v_{p,q}) = \begin{cases} \tilde{U}_{i/2}[q, j] \;\; \forall\, a \in \{0, 1\}, \;\text{ if } i \text{ is even and } p = i+1 \\ 1, \;\text{ if } j = q,\, a = 0,\, i \text{ is odd and } p = i+1 \\ -1, \;\text{ if } j = q,\, a = 1,\, i \text{ is odd and } p = i+1 \\ 0, \;\text{ else} \end{cases}$$

It is clear that for any transition between the $2i^{th}$ and $(2i+1)^{th}$ layer, the transition is query independent and the unitary $U_{2i}^{O_x} = \tilde{U}_i$. This simulates the query-independent unitaries of the circuit C. Meanwhile, the unitary $U_{2i+1}^{O_x}$ is query dependent and is a diagonal matrix where the jth entry takes the value $+1$ if $x_{L(v_{2i+1,j})} = 1$ and -1 otherwise, i.e., the unitary acts as

$$U_{2i+1}^{O_x} \left|v_{2i+1,j}\right\rangle = (-1)^{x_{L(v_{2i+1,j})}} \left|v_{2i+1,j}\right\rangle.$$

We can see that the unitary $U_{2i+1}^{O_x}$ exactly simulates a query to the oracle O_x.

So, the final state after the evolution of the branching program can be given as

$$\left|\psi_f\right\rangle = U_l^{O_x} \cdots U_1^{O_x} \left|v_0\right\rangle = \tilde{U}_{l/2} O_x \tilde{U}_{l/2-1} \cdots \tilde{U}_1 O_x \tilde{U}_0 \left|v_0\right\rangle$$

Which is equivalent to the final state of the circuit C. Finally, on measuring ψ_f and obtaining an output $v_{l,j}$, an input x is accepted iff $v_{l,j} \in F$ or equivalently iff $\left|j\right\rangle \in F_c$. So, the probability of accepting an input x by $\mathcal{G}$ is

$$Pr[accept] = \sum_{v_{l,j} \in F} \left| \left\langle v_{l,j} | \psi_f \right\rangle \right|^2 = \sum_{j \in F_c} \left| \left\langle j \right| C \left|0\right\rangle \right|^2$$

This shows that the $(2^q, 2t+1)$-GQBP $\mathcal{G}$ simulates the query circuit C exactly. ◀

In the above reduction note that the unitaries $U_{2i}^{O_x}$ are query independent. We call the nodes that evolve in a query-independent way as '**dummy**' nodes. In the next lemma, we show that any level with only dummy nodes in a GQBP is removable without any overhead but with a possible reduction in width and length.

▶ **Lemma 20.** *Any level with only dummy nodes in a GQBP is removable without any width or length overhead.*

Proof. Consider a GQBP $\mathcal{G}$ with 3 levels. Without loss of generality, let all levels contain k nodes, and the second level only contains dummy nodes. Let $Q_i = \{v_{i,j} : j \in \{0, 1, \cdots, k-1\}\}$ for $i \in \{0, 2\}$ and $Q_2 = \{d_0, d_1 \cdots, d_{k-1}\}$. Fix an arbitrary input x. Per the definition, the transition matrix $U_1^{O_x}$ from level 1 to level 2 must be unitary. Similar is the case for the transition function $U_2^{O_x}$ for evolution from level 2 to level 3.

Let $U_c^{O_x} = U_2^{O_x} \cdot U_1^{O_x}$. We show that the GQBP $\mathcal{G}$ can be replaced by another GQBP $\mathcal{G}$', which has only two levels. Consider two nodes $v_{0,i}$ and $v_{3,j}$ of $\mathcal{G}$. Then, on considering all the paths from $v_{0,i}$ to $v_{2,j}$, we can get the combined transition amplitude as

$$\sum_{l=0}^{k-1} \delta(d_l, b, v_{2,j})\delta(v_{0,i}, a, d_l) = \sum_{l=0}^{k-1} U_2^{O_x}[j, l]U_1^{O_x}[l, i] = U_c^{O_x}[j, i].$$

See that this amplitude is independent of b and dependent only on a since d_ls are dummy nodes. So, it is possible to construct a transition function that is defined as $\delta'(v_{0,i}, a, v_{2,j}) = \sum_{l=0}^{k-1} \delta(d_l, b, v_{2,j})\delta(v_{0,i}, a, d_l)$ that mimics the transition from the node $v_{0,i}$ to $v_{2,j}$ in $\mathcal{G}$. Since this is true for any two nodes $v_{0,i} \in Q_0$ and $v_{1,i} \in Q_2$, we can replace the GQBP $\mathcal{G}$ with a GQBP $\mathcal{G}$ that has only two levels with Q_0 and Q_2 for nodes and with transition function δ' for transitions between nodes in Q_0 and Q_2. ◀

We can now use Lemma 20 to tighten the result in Theorem 19 to obtain the following corollary.

▶ **Corollary 21.** *For any q-qubit t-query quantum circuit C, there exists an $(2^q, t)$-GQBP that simulates C.*

From this result, one would hope that any $(w, l)-$GQBP would be reducible to a $\log w$-width l-query circuit. However, the GQBP we presented for the $Parity_n$ function serves a a contradiction to this. To see this, note that any query circuit would require at least $\log n$ qubits (that correspond to the indices) that would be used to query the oracle. So, any query circuit computing the $Parity_n$ function exactly will need at least $\log n$ qubits and $n/2$ queries. The lower bound on the number of queries comes from the facts that $Q_E(f) \geq deg(f)/2$ and that $deg(Parity_n) = n$. Now, if the above assumption were true, then clearly, we get a lower bound on w as $w \geq n$. However, the GQBP presented in Section 3 for the $Parity_n$ function has length $l = n/2$ but width 2 that contradicts this lower bound.

We now present the following theorem that characterizes a reduction from a GQBP to a query circuit.

▶ **Theorem 22.** *For any (w, l)-GQBP $\mathcal{G}$, there exists an $(l \log w + \log n + 1)$-qubit $2l$-query circuit C with access to input through an oracle O_x that simulates $\mathcal{G}$ exactly.*

The proof of this theorem is presented in Appendix B.

6 Conclusion and Open Directions

In this work we present a quantum variant of a decision graph [4], commonly known as a branching program, that mimics the evolution of a probabilistic (classical) branching program and further enables it to do so in a superposition. We are primarily concerned about the relationship of quantum query circuits and we prove various results explaining why and how they are equivalent (or, not).

There are several interesting questions that wait to be answered.

What would be the relation between (non-query) quantum circuits or quantum TMs and GQBP, especially with respect to the number of ancillæ qubits of the former and the space complexity of the latter? For example, it is known that polynomial-size classical branching

programs are equivalent to non-uniform logspace classical TMs and poly-size Boolean circuits. Many results about classical branching programs rely on counting arguments and that certainly does not readily work in the quantum case.

We showed that an AQBP is strictly weaker than a GQBP by essentially drawing attention to the fact that the former requires length n to be able to query *every input bit* which the latter does not. This actually stems from the structural requirement of AQBP to query the same variable in a level. Thus, to really compare the strength of the two models, one should consider programs of length $\Omega(n)$, and therefore, of bounded sub-exponential width. We conjecture that a GQBP of length $t = \Omega(n)$ and width $w = O(poly(n))$ is strictly stronger than an AQBP of similar length and width.

It is often assumed that a quantum model can do everything that a classical model can do, and even more. However, that does not appear to hold here. Specifically, it is not clear how to simulate a deterministic classical branching program of width w and length l using a GQBP of width $O(w)$ and length $O(l)$. The difficulty lies in the requirement that the transition function of a GQBP has to be quantumly well-behaved.

The above question is trivially resolved if we instead consider simulating a *Permutation Branching Programs* (PBP). A PBP is a classical levelled branching program that queries the same input bit in a level, and depending upon its value, performs a permutation on the nodes in that level (observe that a permutation matrix is also unitary). An AQBP is a quantum version of a PBP. Barrington's seminal result about NC^1 showed how to simulate any problem in that class using a width-5 (polynomial length) PBP [3]. Improving this result, Ablayebv et al. [2] proved only width 2 is necessary to simulate any NC^1 problem using an AQBP. Based on our understanding of GQBP, we conjecture that a bounded-width GQBP exists for any QNC^1 problem.

References

1. Farid Ablayev, Aida Gainutdinova, and Marek Karpinski. On computational power of quantum branching programs. In *Fundamentals of Computation Theory: 13th International Symposium, FCT 2001 Riga, Latvia, August 22–24, 2001 Proceedings 13*, pages 59–70. Springer, 2001.
2. Farid Ablayev, Cristopher Moore, and Christopher Pollett. Quantum and stochastic branching programs of bounded width: Track a. In *Automata, Languages and Programming: 29th International Colloquium, ICALP 2002 Málaga, Spain, July 8–13, 2002 Proceedings 29*, pages 343–354. Springer, 2002.
3. David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989. doi:10.1016/0022-0000(89)90037-8.
4. William Joseph Masek. A fast algorithm for the string editing problem and decision graph complexity. Master's thesis, Massachusetts Institute of Technology, 1976.
5. Dmitri Maslov, Jin-Sung Kim, Sergey Bravyi, Theodore J Yoder, and Sarah Sheldon. Quantum advantage for computations with limited space. *Nature Physics*, 17(8):894–897, 2021.
6. Masaki Nakanishi, Kiyoharu Hamaguchi, and Toshinobu Kashiwabara. Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction. In *Computing and Combinatorics: 6th Annual International Conference, COCOON 2000 Sydney, Australia, July 26–28, 2000 Proceedings 6*, pages 467–476. Springer, 2000.
7. Alexander A Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *International Symposium on Fundamentals of Computation Theory*, pages 47–60. Springer, 1991.
8. Martin Sauerhoff and Detlef Sieling. Quantum branching programs and space-bounded nonuniform quantum complexity. *Theoretical Computer Science*, 334(1-3):177–225, 2005.
9. Robert Wille, Stefan Hillmich, and Lukas Burgholzer. Decision diagrams for quantum computing. In *Design Automation of Quantum Computers*, pages 1–23. Springer, 2022.

A Proof of Theorem 18

We first proof the reduction from an AQBP to a query circuit under the QRAM model.

Proof. Let the input string be stored in QRAM. At any particular level of a (d, l)-AQBP, the maximum number of nodes is d. At each level, denote each node by a basis state from $|0\rangle$ to $|d-1\rangle$. Then, the total number of qubits required to express the state of a (d, l)-AQBP is $\log(d)$ since after the ith step of evolution, the state would be in a superposition of nodes of level i (we assume the source node is at level 0.)

Next, at iteration i, we perform a quantum transformation of the type $\langle j_i, U_i(0), U_i(1)\rangle$ that takes a superposition of nodes at level $i-1$ to a superposition of nodes at level i. Alternatively, for $a \in \{0, 1\}$, $U_i(a)$ maps each node at level $i-1$ to a superposition of nodes at level i. Since we denote the nodes by computational basis states, let $\widehat{U}_i(a)$ be the unitary that corresponds to the unitary $U_i(a)$ in the computational basis state notation.

Now, construct the circuit C_P that simulates the (d, l)-AQBP P as follows:

1. Initialize the circuit with $\log d$ qubits with the initial state as the state $|0\rangle$ corresponding to the source node v_0.
2. For $i = 1$ to l:
   a. Apply $\widehat{U}_i(0)$.
   b. Controlled on the j_i^{th} qubit in QRAM being 1, apply $\widehat{U}_i^\dagger(0)\widehat{U}_i(1)$.
3. Measure the final state $|\psi_l\rangle$ in the computational basis. If the measurement outcome corresponds to a node in F, then accept the input. Else, reject the input.

We show the correctness of the circuit C_P now. To prove this it suffices to show that the final state obtained just before measurement corresponds to the final state obtained in the AQBP P, i.e, if the circuit outputs the state

$$|\psi_l\rangle = \widehat{U}_l(x_{j_l})\widehat{U}_{l-1}(x_{j_{i-1}})\cdots\widehat{U}_2(x_{j_2})\widehat{U}_1(x_{j_1})\,|\psi_0\rangle$$

The circuit starts in the state $|0\rangle = |\psi_0\rangle$ that corresponds to the node v_0. Next, at iteration i, in step 2 we apply only $\widehat{U}_i(0)$ to $|\psi_{i-1}\rangle$ if x_{j_i} qubit is 0. Else, we apply $\widehat{U}_i(0)\widehat{U}_i^\dagger(0)\widehat{U}_i(1) = \widehat{U}_i(0)$ to $|\psi_{i-1}\rangle$ if $x_{j_i} = 1$. So, the final state $|\psi_l\rangle$ we obtain is of the form,

$$|\psi_l\rangle = \widehat{U}_l(x_{j_l})\widehat{U}_{l-1}(x_{j_{i-1}})\cdots\widehat{U}_2(x_{j_2})\widehat{U}_1(x_{j_1})\,|\psi_0\rangle$$

as expected. Finally, we measure $|\psi_l\rangle$ to see if the measurement outcome lies in the set of basis states that correspond to the accepting set F.

It is straightforward that the total number of qubits used in C_P is $\log d$ and the total number of controlled gates used is at most l. ◀

Next, we prove the reduction from an AQBP to a query circuit under the standard oracle model.

Proof. We make the node to basis state mapping and $U_i(a)$ to $\widehat{U}_i(a)$ mappings for $a \in \{0, 1\}$ as done for the QRAM setup. However, now since the query is made to an oracle O_x that contains the input, at each iteration we also need an index that needs to be queried at that iteration. For this, we need an extra $\log n$ qubits and 1 qubit to store the value of the query. Let S_i denote the $\log n$ qubit unitary that performs the mapping $|0\rangle \longrightarrow |j_i\rangle$. Here we assume the oracle O_x as the standard oracle that acts on a $\log n + 1$ qubits state as $O_x\,|a\rangle\,|b\rangle \longrightarrow |a\rangle\,|b \oplus x_a\rangle$.

Construct the circuit C_P corresponding to the (d, l)-AQBP P as follows.

1. Initialize the circuit with 3 registers $R_1R_2R_3$ of sizes $\log n, \log d$ and 1 respectively with initial states as $|0\rangle$ in all the registers.
2. For $i = 1$ to l:
   a. Apply S_i on R_1 to obtain the state $|j_i\rangle$.
   b. Query the oracle O_x and store the query in R_3.
   c. Controlled on R_3 being in the state $|a\rangle$, apply $\widehat{U}_i(a)$ for $a \in \{0, 1\}$.
   d. Query the oracle O_x to reset R_3 to $|0\rangle$.
   e. Apply $S_i^\dagger$ on R_1 to reset R_1 to $|0\rangle$.
3. Measure R_2 of the final state $|\phi_l\rangle$ in the computational basis. If the measurement outcome corresponds to a node in F, then accept the input. Else, reject the input.

Now, we show the correctness of the circuit C_P. We first initialize the circuit with 3 registers $R_1R_2R_3$, the index register, the node register and the ancilla register, all set to $|0\rangle$. At iteration i, we perform the following operations:

$$\begin{aligned}
|\phi_{i-1}\rangle = |0\rangle |\psi_{i-1}\rangle |0\rangle &\xrightarrow{S_i\otimes\mathbb{I}\otimes\mathbb{I}} |j_i\rangle |\psi_{i-1}\rangle |0\rangle \\
&\xrightarrow{O_x} |j_i\rangle |\psi_{i-1}\rangle |x_{j_i}\rangle \\
&\xrightarrow{\mathbb{I}\otimes\left[\widehat{U}_i(0)\otimes|0\rangle\langle 0|+\widehat{U}_i(1)\otimes|1\rangle\langle 1|\right]} |j_i\rangle \left[\widehat{U}_i(x_{j_i}) |\psi_{i-1}\rangle\right] |x_{j_i}\rangle = |j_i\rangle |\psi_i\rangle |x_{j_i}\rangle \\
&\xrightarrow{O_x} |j_i\rangle |\psi_i\rangle |0\rangle \\
&\xrightarrow{S_i^\dagger\otimes\mathbb{I}\otimes\mathbb{I}} |0\rangle |\psi_i\rangle |0\rangle = |\phi_i\rangle
\end{aligned}$$

So, the state of the R_2 register after i^{th} iteration is

$$|\psi_i\rangle = \widehat{U}_i(x_{j_i}) |\psi_{i-1}\rangle .$$

So, the final state of R_2 register would be of the form

$$|\psi_l\rangle = \widehat{U}_l(x_{j_l})\widehat{U}_{l-1}(x_{j_{i-1}}) \cdots \widehat{U}_2(x_{j_2})\widehat{U}_1(x_{j_1}) |\psi_0\rangle$$

as required. We then measure this register and check if the output is a basis state corresponding to some node in F to accept or reject the input x.

Clearly, the number of queries made to the oracle O_x is $2 \cdot l$ and the total number of qubits used is $\log n + \log d + 1$. ◀

B Proof of Theorem 22

▶ **Theorem 23** (Restated Theorem 22). *For any (w, l)-GQBP $\mathcal{G}$, there exists an $(l \log w + \log n + 1)$-qubit $2l$-query circuit C with access to input through an oracle O_x that simulates $\mathcal{G}$ exactly.*

Proof. Fix an arbitrary width w. We now prove the reduction by induction on the length of GQBP. Let $\mathcal{G} = (Q, E, |v_0\rangle, L, \delta, F)$ be a (w, l)-GQBP. For ease of use we we denote $\delta(v_{i,j}, a, v_{i+1,k})$ as $\beta^{(i+1)}_{j,k,a}$. Let $l = 1$.

From the definition, for the only step of the evolution of this program, we define the unitary $U_1^{O_x}$ as

$$U_1^{O_x} |v\rangle = \sum_{v' \in Q_1} \delta(v, O_x(L(v)), v') |v'\rangle = \sum_{v' \in Q_1} \beta^{(1)}_{v,v',L(v_i)} |v'\rangle .$$

Then, the final state we obtain before the measurement is given by

$$\begin{aligned}
|\psi_1\rangle &= U_1^{O_x} |v_0\rangle \\
&= \sum_{v_i \in Q_0} \alpha_i U_1^{O_x} |v_i\rangle \\
&= \sum_{v_i \in Q_0} \alpha_i \left[\sum_{v_j \in Q_1} \beta^{(1)}_{v_i, v_j, L(v_i)} |v_j\rangle \right] \\
&= \sum_{v_j \in Q_1} \left[\sum_{v_i \in Q_0} \alpha_i \beta^{(1)}_{v_i, v_j, L(v_i)} \right] |v_j\rangle
\end{aligned}$$

So, the probability of obtaining one of the accept states is

$$Pr_{\mathcal{G}}[accept] = \sum_{v_j \in F} \left| \sum_{v_i \in Q_0} \alpha_i \beta^{(1)}_{v_i, v_j, L(v_i)} \right|^2$$

Construct a circuit $\mathcal{C}_{\mathcal{G}}$ with two registers R_1, R_2 each of size $\lceil w \rceil$ qubits, one register R_3 of size $\log n$ and one register R_4 of size 1 qubit all initialized to $|0\rangle$. Let $\mathcal{C}_{\mathcal{G}}$ be provided with an access to the input X through the oracle O_X and let $\widehat{U}_L$ be the unitary that acts as $\widehat{U}_L |j\rangle |0\rangle = |j\rangle |L(v_{0,j})\rangle$. Since we know that initial state of $\mathcal{G}$, we can construct a unitary U_{init} that act as $U_{init} |0\rangle = \sum_{i \in [w]} \alpha_i |i\rangle$ apriori. Then one step of $\mathcal{G}$ can be simulated in $\mathcal{C}_{\mathcal{G}}$ as follows:

$$\begin{aligned}
|0\rangle |0\rangle |0\rangle |0\rangle &\xrightarrow{U_{init}} \sum_{i \in [w]} \alpha_i |i\rangle |0\rangle |0\rangle |0\rangle \\
&\xrightarrow{CX(R_1,R_2)} \sum_{i \in [w]} \alpha_i |i\rangle |i\rangle |0\rangle |0\rangle \\
&\xrightarrow{\widehat{U}_L(R_1,R_3)} \sum_{i \in [w]} \alpha_i |i\rangle |i\rangle |L(v_{0,i})\rangle |0\rangle \\
&\xrightarrow{O_X(R_3,R_4)} \sum_{i \in [w]} \alpha_i |i\rangle |i\rangle |L(v_{0,j})\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{C-U_1(R_1,R_2,R_4)} \sum_{i \in [w]} \alpha_i |i\rangle \Big(\sum_{j \in [w]} \beta^{(1)}_{i,j,X_i} |j\rangle \Big) |L(v_{0,j})\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{O_X(R_3,R_4)} \sum_{i \in [w]} \alpha_i |i\rangle \Big(\sum_{j \in [w]} \beta^{(1)}_{i,j,X_i} |j\rangle \Big) |L(v_{0,j})\rangle |0\rangle \\
&\xrightarrow{\widehat{U}_L(R_1,R_3)} \sum_{i \in [w]} \alpha_i |i\rangle \Big(\sum_{j \in [w]} \beta^{(1)}_{i,j,X_i} |j\rangle \Big) |0\rangle |0\rangle \\
&= \sum_{j \in [w]} \Big(\sum_{i \in [w]} \alpha_i \beta^{(1)}_{i,j,X_i} |i\rangle \Big) |j\rangle |0\rangle |0\rangle = |\psi_f\rangle \ \text{(say)}
\end{aligned}$$

Here, $C - U_1$ is defined as

$$C - U_1 = \sum_{i \in [w]} |i\rangle \langle i| \otimes \Big[\sum_{j \in [w]} \beta^{(1)}_{i,j,0} |j\rangle \langle i| \otimes |0\rangle \langle 0| + \sum_{j \in [w]} \beta^{(1)}_{i,j,1} |j\rangle \langle i| \otimes |1\rangle \langle 1| \Big].$$

Note that the unitary $C - U_1$ is query independent. Next, let $\mathcal{M} = P_0, P_1$ be a projective measurement where $P_1 = \sum_{v_i \in F} |i\rangle \langle i|$ and $P_0 = I - P_1$. Once we measure the second register using $\mathcal{M}$,the probability of accepting X can be computed as

$$Pr_{\mathcal{C}_\mathcal{G}}[accept] = \sum_{v_j \in F} \left| \sum_{i \in [w]} \alpha_i \beta^{(1)}_{i,j,X_i} |i\rangle \right|^2 = \sum_{v_j \in F} \left| \sum_{i \in [w]} \alpha_i \beta^{(1)}_{i,j,X_i} \right|^2.$$

Note that this is exactly the probability of accepting the input X in $\mathcal{G}$. This gives that the $\log(w) + \log(n) + 1$-qubit 2-query $\mathcal{C}_\mathcal{G}$ simulates the $(w, 1)$-GQBP $\mathcal{G}$ exactly.

Now, assume the result true for some $l = t$. Let $\mathcal{G}$'$= (Q', E', |v'_0\rangle, L', \delta', F')$ be a $(w, t+1)$-GQBP. Let the state at the end of t layers be given as

$$|\psi_t\rangle = \sum_{v_i \in Q_t} \gamma_i |v_i\rangle .$$

The transition unitary $U_t^{O_x}$ of the last step of the evolution of $\mathcal{G}$' can be defined as

$$U_t^{O_x} |v\rangle = \sum_{v' \in Q'_{t+1}} \delta'(v, O_x(L'(v)), v') |v'\rangle = \sum_{v' \in Q'_{t+1}} \beta^{(t)}_{v,v',L'(v_i)} |v'\rangle .$$

Then, the final state of the GQBP would be

$$|\psi_{t+1}\rangle = U_t^{O_x} |\psi_t\rangle = \sum_{v_i \in Q_t} \gamma_j \left[\sum_{v_j \in Q_{t+1}} \beta^{(t)}_{v,v_j,L'(v_i)} |v_j\rangle \right] = \sum_{v_j \in Q_{t+1}} \left[\sum_{v_i \in Q_t} \gamma_i \beta^{(t)}_{v,v_j,L'(v_i)} \right] |v_j\rangle .$$

So, the probability of accepting an input X is

$$Pr_{\mathcal{G}'}[accept] = \sum_{v_j \in F'} \left| \sum_{v_i \in Q_t} \gamma_i \beta^{(t)}_{v_i,v_j,L(v_i)} \right|^2$$

Next, let $\mathcal{C}'_\mathcal{G}$ be the circuit that computes the first t layers of $\mathcal{G}$' exactly using $t \log(w) + \log(n) + 1$ qubits and $2t$ queries. Then, the final state of this circuit will be

$$\left|\psi_t^C\right\rangle = \sum_{i \in [w]} \gamma_i |\chi_i\rangle |i\rangle |0\rangle |0\rangle$$

for some normalized states $|\chi_i\rangle$ of $(t-1)\log(w)$ qubits. Now, append a register of $\log(w)$ qubits initialized to $|0\rangle$ to the circuit $\mathcal{C}'_\mathcal{G}$. For ease of computation, we append it in between the current second and the third register. So, the new state of the circuit will be

$$\left|\psi_t^C\right\rangle = \sum_{j \in [w]} \gamma_i |\chi_i\rangle |i\rangle |0\rangle |0\rangle |0\rangle = R_1 R_2 R_3 R_4 R_5 \text{ (say)}.$$

Now, we simulate the final step of $\mathcal{G}$ as below:

$$
\begin{aligned}
\left|\psi_t^C\right\rangle &= \sum_{j\in[w]} \gamma_i \left|\chi_i\right\rangle \left|i\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle \\
&\xrightarrow{CX(R_2,R_3)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \left|i\right\rangle \left|0\right\rangle \left|0\right\rangle \\
&\xrightarrow{\widehat{U}_L(R_2,R_4)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \left|i\right\rangle \left|L(v_{0,i})\right\rangle \left|0\right\rangle \\
&\xrightarrow{O_X(R_4,R_5)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \left|i\right\rangle \left|L(v_{0,j})\right\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{C-U_t(R_2,R_3,R_5)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \Big(\sum_{j\in[w]} \beta^{(t)}_{i,j,X_i} \left|j\right\rangle \Big) \left|L(v_{0,j})\right\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{O_X(R_4,R_5)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \Big(\sum_{j\in[w]} \beta^{(t)}_{i,j,X_i} \left|j\right\rangle \Big) \left|L(v_{0,j})\right\rangle \left|0\right\rangle \\
&\xrightarrow{\widehat{U}_L(R_2,R_4)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \Big(\sum_{j\in[w]} \beta^{(t)}_{i,j,X_i} \left|j\right\rangle \Big) \left|0\right\rangle \left|0\right\rangle \\
&= \sum_{j\in[w]} \Big(\sum_{i\in[w]} \gamma_i \beta^{(t)}_{i,j,X_i} \left|\chi_j\right\rangle \left|i\right\rangle \Big) \left|j\right\rangle \left|0\right\rangle \left|0\right\rangle = \left|\psi_f\right\rangle \ \text{(say)}
\end{aligned}
$$

Similar to the length 1 case, $C - U_t$ is defined as

$$
C - U_t = \sum_{i\in[w]} \left|i\right\rangle \left\langle i\right| \otimes \Big[\sum_{j\in[w]} \beta^{(t)}_{i,j,0} \left|j\right\rangle \left\langle i\right| \otimes \left|0\right\rangle \left\langle 0\right| + \sum_{j\in[w]} \beta^{(t)}_{i,j,0} \left|j\right\rangle \left\langle i\right| \otimes \left|1\right\rangle \left\langle 1\right| \Big].
$$

Then finally on measuring the third register, we get that the probability of accepting an input X is

$$
Pr_{\mathcal{C}'_{\mathcal{G}}}[accept] = \sum_{v_j\in F'} \Big| \sum_{i\in[w]} \gamma_i \beta^{(t)}_{i,j,X_i} \left|i\right\rangle \Big|^2 = \sum_{v_j\in F'} \Big| \sum_{i\in[w]} \gamma_i \beta^{(t)}_{i,j,X_i} \Big|^2 .
$$

which equals the probability of accepting the input X in $\mathcal{G}$'. So, we obtain a $((l+1)\log(w) + \log(n) + 1)$-qubit $2l+2$-query circuit to simulate $\mathcal{G}$' exactly as required. ◀

Tight Bounds for the Randomized and Quantum Communication Complexities of Equality with Small Error

Olivier Lalonde
DIRO, Université de Montréal, Canada

Nikhil S. Mande[1]
University of Liverpool, UK

Ronald de Wolf
QuSoft, CWI and University of Amsterdam, The Netherlands

Abstract

We investigate the randomized and quantum communication complexities of the well-studied Equality function with small error probability ε, getting the optimal constant factors in the leading terms in various different models.

The following are our results in the *randomized* model:

- We give a general technique to convert public-coin protocols to private-coin protocols by incurring a small multiplicative error at a small additive cost. This is an improvement over Newman's theorem [Inf. Proc. Let.'91] in the dependence on the error parameter.
- As a consequence we obtain a $(\log(n/\varepsilon^2) + 4)$-cost private-coin communication protocol that computes the n-bit Equality function, to error ε. This improves upon the $\log(n/\varepsilon^3) + O(1)$ upper bound implied by Newman's theorem, and matches the best known lower bound, which follows from Alon [Comb. Prob. Comput.'09], up to an additive $\log\log(1/\varepsilon) + O(1)$.

The following are our results in various *quantum* models:

- We exhibit a one-way protocol with $\log(n/\varepsilon) + 4$ qubits of communication for the n-bit Equality function, to error ε, that uses only pure states. This bound was implicitly already shown by Nayak [PhD thesis'99].
- We give a near-matching lower bound: any ε-error one-way protocol for n-bit Equality that uses only pure states communicates at least $\log(n/\varepsilon) - \log\log(1/\varepsilon) - O(1)$ qubits.
- We exhibit a one-way protocol with $\log(\sqrt{n}/\varepsilon) + 3$ qubits of communication that uses *mixed* states. This is tight up to additive $\log\log(1/\varepsilon) + O(1)$, which follows from Alon's result.
- We exhibit a one-way entanglement-assisted protocol achieving error probability ε with $\lceil\log(1/\varepsilon)\rceil + 1$ classical bits of communication and $\lceil\log(\sqrt{n}/\varepsilon)\rceil + 4$ shared EPR-pairs between Alice and Bob. This matches the communication cost of the classical public coin protocol achieving the same error probability while improving upon the amount of prior entanglement that is needed for this protocol, which is $\lceil\log(n/\varepsilon)\rceil + O(1)$ shared EPR-pairs.

Our upper bounds also yield upper bounds on the approximate rank, approximate nonnegative-rank, and approximate psd-rank of the Identity matrix. As a consequence we also obtain improved upper bounds on these measures for a function that was recently used to refute the randomized and quantum versions of the log-rank conjecture (Chattopadhyay, Mande and Sherif [J. ACM'20], Sinha and de Wolf [FOCS'19], Anshu, Boddu and Touchette [FOCS'19]).

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Quantum complexity theory

Keywords and phrases Communication complexity, quantum communication complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.32

[1] Work done while the author was a postdoc at CWI, Amsterdam, and supported by the Dutch Research Council (NWO) through QuantERA ERA-NET Cofund project QuantAlgo (project number 680-91-034).

© Olivier Lalonde, Nikhil S. Mande, and Ronald de Wolf;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 32; pp. 32:1–32:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version *Previous Version*: https://arxiv.org/abs/2107.11806

Funding *Olivier Lalonde*: Supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).
Ronald de Wolf: Partially supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.037), and through QuantERA ERA-NET Cofund project QuantAlgo (680-91-034).

Acknowledgements We thank Troy Lee, Ignacio Villanueva, and Zhaohui Wei for early discussions related to the result of Section 4.3. We thank Swagato Sanyal for discussions at an early stage of this work, from which the question of pinning down the exact communication complexity of Equality for small error arose.

1 Introduction

Yao [24] introduced the classical model of communication complexity, and also subsequently introduced its quantum analogue [25]. Communication complexity has important applications in several disciplines, such as lower bounds on circuits, data structures, streaming algorithms, and many other areas (see, for example, [11, 20] and the references therein). The basic model of communication complexity involves two parties, usually called Alice and Bob, who wish to jointly compute $F(x, y)$ for a known function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where Alice holds $x \in \{0,1\}^n$ and Bob holds $y \in \{0,1\}^n$. The parties use a communication protocol agreed upon in advance to compute $F(x, y)$. They are individually computationally unbounded and the cost is the amount of communication between the parties on the worst-case input.

Consider the n-bit *Equality* function, denoted $\mathsf{EQ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ (or simply EQ when n is clear from context), and defined as $\mathsf{EQ}_n(x, y) = 1$ iff $x = y$. This is arguably the simplest and most basic problem in communication complexity. It is well known that its deterministic communication complexity equals n, which is maximal. However, Yao [24] already showed that if we allow some small constant error probability, then the communication complexity becomes much smaller. In this paper we pin down the small-error communication complexity of *Equality* in various communication models. Our bounds are essentially optimal both in terms of n and in terms of the error. While our optimal upper bounds only give small improvements over known bounds, Equality is such a fundamental communication problem that we feel it is worthwhile to pin down its complexity as precisely as possible and to find protocols that are as efficient as possible.

1.1 Prior work

Given a function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, define the $2^n \times 2^n$ *communication matrix* of F, denoted M_F, by $M_F(x, y) = F(x, y)$. Define the *ε-approximate rank* of a matrix M, denoted $\mathsf{rk}_\varepsilon(M)$, to be the minimum number of rank-1 matrices needed such that their sum is ε-close to M entrywise (equivalently, $\mathsf{rk}_\varepsilon(M)$ is the minimum rank among all matrices that are ε-close to M entrywise). If the rank-1 matrices are additionally constrained to be entrywise nonnegative, then the resulting measure is called the *ε-approximate nonnegative-rank* of M, denoted $\mathsf{rk}^+_\varepsilon(M)$. By definition, $\mathsf{rk}^+_\varepsilon(M_F) \geq \mathsf{rk}_\varepsilon(M_F)$. Denote ε-error randomized communication complexity by $\mathsf{R}^{\mathsf{pri}}_\varepsilon(\cdot)$ when the players have access to private randomness, and $\mathsf{R}^{\mathsf{pub}}_\varepsilon(F)$ when the players have access to public randomness (i.e., shared coin flips). Let $\mathsf{Q}^{\mathsf{pri}}_\varepsilon(\cdot)$ denote ε-error quantum communication complexity, assuming private randomness. In all quantum communication models under consideration in this paper, except for the last one, Alice and Bob do not have access to pre-shared entanglement.

Krause [9] showed the following lower bound on the randomized communication complexity of a Boolean function in terms of the approximate nonnegative-rank of its communication matrix.

▶ **Theorem 1** ([9]). *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function and $\varepsilon > 0$. Then,*

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon}(F) \geq \log \mathsf{rk}^{+}_{\varepsilon}(M_F).$$

Analogous to this, the following lower bound is known on the quantum communication complexity of a Boolean function, due to Nielsen [18] and Buhrman and de Wolf [5].

▶ **Theorem 2** ([18, 5]). *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $\varepsilon > 0$. Then,*

$$\mathsf{Q}^{\mathsf{pri}}_{\varepsilon}(F) \geq \frac{1}{2} \log \mathsf{rk}_{\varepsilon}(M_F).$$

A similar proof as that of [5] can be used to show that the quantum communication complexity of a Boolean function is bounded below by the logarithm of its *approximate psd-rank*, which we define below. Let M be a matrix with nonnegative real entries. A rank-d *psd-factorization* of M consists of a set of $d \times d$ complex[2] psd matrices A_i (one for each row of M) and B_j (one for each column of M), such that for all i, j we have $M_{ij} = \mathrm{tr}(A_i B_j)$. The *psd-rank* of M, denoted $\mathsf{rk}^{\mathrm{psd}}(M)$, is the minimal d for which M has such a psd factorization. This notion has gained a lot of interest in areas such as semidefinite optimization, communication complexity, and others. See Fawzi et al. [7] for an excellent survey. The *ε-approximate psd-rank* of M, which we denote by $\mathsf{rk}^{\mathrm{psd}}_{\varepsilon}(M)$, is the minimum psd-rank among all matrices that are ε-close to M entrywise.

▶ **Theorem 3.** *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $\varepsilon > 0$. Then,*

$$\mathsf{Q}^{\mathsf{pri}}_{\varepsilon}(F) \geq \log \mathsf{rk}^{\mathrm{psd}}_{\varepsilon}(M_F) + 1.$$

For completeness, we prove this in Appendix A. It is easy to show that $\mathsf{rk}^{\mathrm{psd}}_{\varepsilon}(M_F) \leq \mathsf{rk}^{+}_{\varepsilon}(M_F)$. Alon [1] showed the following bounds on the approximate rank of the Identity matrix.

▶ **Theorem 4** ([1]). *There exists a positive constant c such that the following holds for all integers $n > 0$ and $1/2^{n/2} \leq \varepsilon \leq 1/4$. Let I denote the $2^n \times 2^n$ Identity matrix. Then,*

$$\mathsf{rk}_{\varepsilon}(I) \geq \frac{cn}{\varepsilon^2 \log\left(\frac{1}{\varepsilon}\right)}.$$

Note that the $2^n \times 2^n$ Identity matrix is the communication matrix of the n-bit Equality function. Theorems 1 and 4 thus imply that for $1/2^{n/2} \leq \varepsilon \leq 1/4$,

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon}(\mathsf{EQ}_n) \geq \log\left(\frac{n}{\varepsilon^2}\right) - \log\log\left(\frac{1}{\varepsilon}\right) - O(1). \tag{1}$$

Newman [16] proved the following theorem that shows that public-coin protocols can be converted to private-coin protocols with an additive error, with a small additive cost in the communication. For the following form, see for example, [11, Claim 3.14].

[2] Often this definition is restricted to real matrices. This can change the psd-rank by a constant factor, but no more than that [12, Section 3.3].

▶ **Theorem 5** (cf. [11, Claim 3.14]). *Let $F : \{0,1\}^n \times \{0,1\}^n$ be a Boolean function. For every $\delta > 0$ and every $\varepsilon > 0$,*

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon+\delta}(F) \leq \mathsf{R}^{\mathsf{pub}}_{\varepsilon}(F) + \log\left(\frac{n}{\delta^2}\right) + O(1).$$

Relabeling variables, Theorem 5 is equivalent to

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon(1+\delta)}(F) \leq \mathsf{R}^{\mathsf{pub}}_{\varepsilon}(F) + \log\left(\frac{n}{\varepsilon^2\delta^2}\right) + O(1).$$

1.2 Our results

In this section we list our results, first those for randomized communication complexity, and then those for quantum communication complexity.

1.2.1 Randomized communication complexity

We give an improved version of Newman's theorem (Theorem 5), which allows us to convert a public-coin protocol to a private-coin one with an optimal dependence on the error. Our proof follows along similar lines as that of Newman's. Our key deviation is that we use a multiplicative form of the Chernoff bound, where previously an additive version was used.

▶ **Theorem 6.** *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. For all $\varepsilon \in [0, 1/2)$ and all $\delta \in (0, 1]$,*

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon(1+\delta)}(F) \leq \mathsf{R}^{\mathsf{pub}}_{\varepsilon}(F) + \log\left(\frac{n}{\varepsilon}\right) + \log\left(\frac{6}{\delta^2}\right).$$

To compare Theorem 5 and Theorem 6, consider the $(1/n)$-error private-coin randomized communication complexity of EQ_n. The ε-error *public-coin* communication complexity of EQ_n is at most $\log(1/\varepsilon)$ (and this can be shown to be tight up to an additive constant). Thus, Theorem 5 can at best give an upper bound of

$$\mathsf{R}^{\mathsf{pri}}_{1/n}(\mathsf{EQ}_n) \leq \log n + \log(n^3) + O(1) = 4\log n + O(1).$$

Equation (1) implies a non-matching lower bound $\mathsf{R}^{\mathsf{pri}}_{1/n}(\mathsf{EQ}_n) \geq 3\log n - \log\log n - O(1)$. On the other hand, Theorem 6 implies a tight upper bound (up to the additive $\log\log n + O(1)$ term) of $3\log n + O(1)$ on the $(1/n)$-error private-coin communication complexity of EQ_n, by converting the $\log(1/\varepsilon)$-cost public-coin protocol for EQ_n to a private-coin protocol.

▶ **Theorem 7.** *For all positive integers $n > 0$ and for all $\varepsilon \in [0, 1/2)$,*

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon}(\mathsf{EQ}) \leq \log\left(\frac{n}{\varepsilon^2}\right) + 4.$$

This shows that Alon's theorem (Theorem 4) is tight up to the $O(\log(1/\varepsilon))$ factor, not only for approximate rank, but also for communication complexity. Theorem 7 and Theorem 1 also imply that the approximate-rank lower bound in Theorem 4 is nearly tight even restricting to *nonnegative* approximations to the Identity matrix.

▶ **Corollary 8.** *Let $n > 0$ be an integer, and let I denote the $2^n \times 2^n$ Identity matrix. Then for all $\varepsilon \in [0, 1/2)$,*

$$\mathsf{rk}^{+}_{\varepsilon}(I) \leq \frac{16n}{\varepsilon^2}.$$

To compare the performance of Theorem 5 with that of Theorem 6 in a more general setting, we consider the natural problem of converting a public-coin protocol to a private-coin protocol while allowing the error to double. Setting $\delta = \varepsilon$ in Theorem 5 and relabeling parameters, we obtain

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon}(F) \leq \mathsf{R}^{\mathsf{pub}}_{\varepsilon/2}(F) + \log\left(\frac{n}{\varepsilon^2}\right) + O(1).$$

However, Theorem 6 yields the following improved dependence on ε by setting $\delta = 1$ and relabeling parameters.

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon}(F) \leq \mathsf{R}^{\mathsf{pub}}_{\varepsilon/2}(F) + \log\left(\frac{n}{\varepsilon}\right) + 4.$$

1.2.2 Quantum communication complexity

Prior to this work, the best known lower bound on the ε-error quantum communication complexity of Equality was $\Omega(\log(n/\varepsilon))$ [5, Proposition 3], with a constant hidden in the $\Omega(\cdot)$ that is less than $1/2$. Theorem 2 and Theorem 4 imply that

$$\mathsf{Q}^{\mathsf{pri}}_{\varepsilon}(\mathsf{EQ}_n) \geq \log\left(\frac{\sqrt{n}}{\varepsilon}\right) - \log\log\left(\frac{1}{\varepsilon}\right) - O(1). \tag{2}$$

In terms of upper bounds, we exhibit a *one-way* quantum communication upper bound with an optimal dependence on ε, that uses only pure-state messages (and hence does not use even private randomness). In particular, by choosing ε to be an arbitrary small polynomial in the input size, this implies that the factor of $1/2$ in Theorem 2 cannot be improved when $F = \mathsf{EQ}_n$. Let $\mathsf{Q}^{\mathrm{pure},\rightarrow}_{\varepsilon}(F)$ be the ε-error quantum communication complexity of F, where the protocols are one-way and Alice is only allowed to send a pure state to Bob. We show the following.

▶ **Theorem 9.** *For all positive integers $n > 0$ and for all $\varepsilon \in [0, 1/2)$,*

$$\mathsf{Q}^{\mathrm{pure},\rightarrow}_{\varepsilon}(\mathsf{EQ}_n) \leq \log\left(\frac{n}{\varepsilon}\right) + 4.$$

The proof uses the probabilistic method to analyze random linear codes. Nayak [15] already used the same upper bound technique to show an upper bound on the bounded-error one-way quantum communication complexity of EQ_n. They did not explicitly derive this error-dependence, but it follows immediately from their construction by plugging in codes with length $O(n/\varepsilon)$ and relative distance $1/2 - \sqrt{\varepsilon}$ in [15, pp.16–17]. We also show that this is nearly tight:

▶ **Theorem 10.** *There exists an absolute constant c such that the following holds. For all positive integers $n > 0$ and for all $\varepsilon \in [1/2^n, 1/4]$,*

$$\mathsf{Q}^{\mathrm{pure},\rightarrow}_{\varepsilon}(\mathsf{EQ}_n) \geq \log\left(\frac{n}{\varepsilon}\right) - \log\log\left(\frac{1}{\varepsilon}\right) - c.$$

While the pure-state protocol of Theorem 9 has optimal dependence on ε (up to the additive $\log\log(1/\varepsilon)$ term), it does not match the n-dependence of the lower bound of Equation (2); in fact, one-way pure-state protocols cannot match this (Theorem 10). However, if we allow one-way *mixed-state* messages, then we can give a better upper bound and close the gap:

▶ **Theorem 11.** *For all positive integers $n > 0$ and for all $\varepsilon \in [0, 1/2)$,*

$$\mathsf{Q}_\varepsilon^{\mathsf{pri}}(\mathsf{EQ}_n) \leq \log\left(\frac{\sqrt{n}}{\varepsilon}\right) + 3.$$

An upper bound of $\log\sqrt{n} + O(1)$ was already proved by Winter [22] for the case of constant ε, and here we obtain the correct dependence also for subconstant ε. Our proof is again probabilistic, using known concentration properties of overlaps of random projectors to allow us to show the existence of appropriate mixed-state messages for Alice and appropriate measurements for Bob. Theorems 3 and 11 also imply upper bounds on the ε-approximate psd-rank of the Identity matrix.

▶ **Corollary 12.** *Let $n > 0$ be an integer, and let I denote the $2^n \times 2^n$ Identity matrix. Then for all $\varepsilon \in [0, 1/2)$,*

$$\mathsf{rk}_\varepsilon^{\mathrm{psd}}(I) \leq \frac{4\sqrt{n}}{\varepsilon}.$$

As noted by Lee, Wei and de Wolf [12, Theorem 17], Alon's approximate rank lower bound (Theorem 4) almost immediately gives a lower bound of $\mathsf{rk}_\varepsilon^{\mathrm{psd}}(I) = \Omega\left(\frac{\sqrt{n}}{\varepsilon\sqrt{\log(1/\varepsilon)}}\right)$. This shows that our upper bound in Corollary 12 is tight up to a multiplicative $O(\sqrt{\log(1/\varepsilon)})$ factor.

We may also consider the amount of entanglement needed to compute EQ_n in the entanglement-assisted setting, where Alice and Bob send classical bits but share an arbitrary input-independent state $|\psi\rangle$ at the start of the protocol, for instance many EPR-pairs. Since entanglement may be used to generate shared randomness by measuring, the classical public-coin protocol yields an entanglement-assisted protocol using $\lceil \log 1/\varepsilon \rceil + 1$ bits of communication and $\lceil \log n/\varepsilon \rceil + O(1)$ shared EPR-pairs. We improve on the amount of shared entanglement that's needed by showing:

▶ **Theorem 13.** *For all positive integers $n > 0$ and for all $\varepsilon \in (0, 1/2)$, there exists a one-way protocol for EQ_n with error probability at most ε using $\lceil \log 1/\varepsilon \rceil + 1$ bits of communication and $\lceil \log \sqrt{n}/\varepsilon \rceil + 4$ shared EPR-pairs.*

We do not know if the amount of classical communication used by our protocol to achieve error probability ε is essentially optimal. Nor do we know if the number of EPR-pairs can be reduced further given this amount of communication.

2 Preliminaries

All logarithms in this paper are taken to base 2. We use $\exp(x)$ to denote e^x, where e denotes Euler's number. For strings $x, y \in \{0,1\}^n$, define their Hamming distance by $d(x, y) := |\{i \in [n] : x_i \neq y_i\}|$. For an event X, let $I(X) \in \{0, 1\}$ denote the *indicator* of X, which is 1 iff X occurs.

▶ **Definition 14** (Linear code)**.** *For integers $N \geq n$, a linear code is a linear function $C : \{0,1\}^n \to \{0,1\}^N$.*

One may view a linear code as an $N \times n$ matrix M over $\mathbb{F}_2$; an input $x \in \{0,1\}^n$ is mapped to N-bit codeword Mx (where the matrix product is taken over $\mathbb{F}_2$). Choosing a random linear code corresponds to choosing an M with uniformly random binary entries.

We use the following well-known multiplicative form of the Chernoff bound [14, Theorem 4.4].

▶ **Lemma 15.** *Let $Z_1, \ldots, Z_n$ be independent random variables taking values in $\{0,1\}$. Let $Z = \sum_{i=1}^n Z_i$, and let $\mu = \mathbb{E}[Z]$. Then for all $\delta \in [0,1]$,*

$$\Pr[Z \geq (1+\delta)\mu] \leq \exp(-\delta^2\mu/3),$$
$$\Pr[Z \leq (1-\delta)\mu] \leq \exp(-\delta^2\mu/2).$$

We refer the reader to [11, 20] for the basics of classical communication complexity, to [19] for quantum computing, and to [23] for an introduction to quantum communication complexity.

3 An improved form of Newman's theorem

Proof of Theorem 6. Let Π be a public-coin protocol that computes F with error ε. Assume without loss of generality that all the random coins are tossed at the beginning of the protocol. That is, for every $x, y \in \{0,1\}^n$,

$$\Pr_r[\Pi(x,y,r) \neq F(x,y)] \leq \varepsilon. \tag{3}$$

Set

$$B = \frac{6n}{\delta^2\varepsilon} \tag{4}$$

and independently choose random strings $r_1, \ldots, r_B$ according to the same distribution as used by Π. For two strings $x, y \in \{0,1\}^n$ and an index $j \in [B]$, let $I_{j,x,y}$ denote the indicator event of r_j being a "bad random string" for x, y:

$$I_{j,x,y} := \begin{cases} 1 & \Pi(x,y,r_j) \neq f(x,y) \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Fix two arbitrary strings $x, y \in \{0,1\}^n$. Equation (3) implies $\Pr_{r_1,\ldots,r_B,j\in[B]}[I_{j,x,y} = 1] \leq \varepsilon$. By linearity of expectation and our choice of B in Equation (4),

$$\mathbb{E}_{r_1,\ldots,r_B}\left[\sum_{j\in[B]} I_{j,x,y}\right] \leq B\varepsilon = \frac{6n}{\delta^2}.$$

We now give an upper bound on $\Pr_{r_1,\ldots,r_B}\left[\sum_{j\in[B]} I_{j,x,y} \geq B\varepsilon(1+\delta)\right]$. Assume without loss of generality that $\Pr_{r_1,\ldots,r_B,j\in[B]}[I_{j,x,y} = 1] = \varepsilon$, and hence $\mathbb{E}_{r_1,\ldots,r_B}\left[\sum_{j\in[B]} I_{j,x,y}\right] = B\varepsilon$ (since the desired probability could only be smaller otherwise). By a Chernoff bound (Lemma 15),

$$\Pr_{r_1,\ldots,r_B}\left[\sum_{j\in[B]} I_{j,x,y} \geq B\varepsilon(1+\delta)\right] \leq \exp\left(-\frac{\delta^2 \cdot 6n}{3\delta^2}\right) = \exp(-2n) < 2^{-2n}.$$

By a union bound over all $x, y \in \{0,1\}^n$,

$$\begin{aligned}
&\Pr_{r_1,\ldots,r_B}\left[\sum_{j\in[B]} I_{j,x,y} \geq B\varepsilon(1+\delta) \text{ for some } x,y \in \{0,1\}^n\right] \\
&\quad \leq \sum_{x,y\in\{0,1\}^n} \Pr_{r_1,\ldots,r_B}\left[\sum_{j\in[B]} I_{j,x,y} \geq B\varepsilon(1+\delta)\right] \\
&\quad < 2^{2n} \cdot 2^{-2n} = 1.
\end{aligned}$$

Hence there exists a choice of $r_1, \ldots, r_B$ such that the following holds for all $x, y \in \{0,1\}^n$:

$$\sum_{j \in [B]} I_{j,x,y} < B\varepsilon(1+\delta). \tag{6}$$

Fixing this choice of $r_1, \ldots, r_B$, Protocol 1 gives a private-coin protocol for F.

Protocol 1 A private-coin protocol for F.

1. Alice samples $j \in [B]$ uniformly at random, and sends it to Bob.
2. Alice and Bob perform the public-coin protocol Π assuming r_j was the public random string.

To show the correctness of this protocol, our choice of B (Equation (4)) and Equations (5) and (6) imply that for all $x, y \in \{0,1\}^n$,

$$\Pr_{j \in [B]}[\Pi(x, y, r_j) \neq f(x, y)] < \frac{B\varepsilon(1+\delta)}{B} = \varepsilon(1+\delta).$$

Hence the protocol has error probability less than $\varepsilon(1+\delta)$. The cost of the first step of the protocol is $\log B$, and the cost of the second step is at most $\mathsf{R}^{\mathsf{pub}}_{\varepsilon}(F)$. Thus, we have,

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon(1+\delta)}(F) \leq \mathsf{R}^{\mathsf{pub}}_{\varepsilon}(F) + \log B = \mathsf{R}^{\mathsf{pub}}_{\varepsilon}(F) + \log \frac{6n}{\delta^2 \varepsilon} = \mathsf{R}^{\mathsf{pub}}_{\varepsilon}(F) + \log \frac{n}{\varepsilon} + \log \frac{6}{\delta^2}.$$

Note that if Π was a one-way protocol, then Protocol 1 is a one-way private-coin protocol. ◀

4 Communication complexity upper bounds

In this section we show randomized and quantum communication upper bounds for Equality.

4.1 Randomized upper bound

As an application of Theorem 6, we recover an optimal small-error private-coin communication complexity upper bound for EQ_n from a naive public-coin protocol of cost $\log(2/\varepsilon)$ and error $\varepsilon/2$:

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon}(\mathsf{EQ}_n) \leq \log\left(\frac{2}{\varepsilon}\right) + \log\left(\frac{n}{\varepsilon}\right) + 3 = \log\left(\frac{n}{\varepsilon^2}\right) + 4. \tag{7}$$

This proves Theorem 7. In contrast, Newman's theorem (Theorem 5) would only give an upper bound of

$$\mathsf{R}^{\mathsf{pri}}_{\varepsilon}(\mathsf{EQ}_n) \leq \log\left(\frac{2}{\varepsilon}\right) + \log\left(\frac{n}{\varepsilon^2}\right) + O(1) = \log\left(\frac{n}{\varepsilon^3}\right) + O(1).$$

In particular, for $\varepsilon = 1/n$ we improve the upper bound from $4\log n + O(1)$ to $3\log n + O(1)$, which turns out to be essentially optimal.

4.2 Quantum upper bound with only pure states

We require the following property of random linear codes.

▷ Claim 16. Let n be a positive integer and let $\delta > 0$. Let $x \neq y \in \{0,1\}^n$ be two arbitrary but fixed strings. Let $N = 4n/\delta^2$. Let $C : \{0,1\}^n \to \{0,1\}^N$ be a random linear code. Then

$$\Pr_C \left[\frac{d(C(x), C(y))}{N} \notin \left[\frac{1}{2} - \delta, \frac{1}{2} + \delta \right] \right] < 2^{-2n}.$$

Proof of Claim 16. For each $i \in [N]$, the random variable $Z_i := I[C(x)_i = C(y)_i]$ equals 1 with probability 1/2 and 0 with probability 1/2. Further, Z_i and Z_j are independent for all $i \neq j \in [N]$. Define $Z = \sum_{i=1}^N Z_i = d(C(x), C(y))$. We have $\mathbb{E}[Z] = N/2$. By a Chernoff bound (Lemma 15),

$$\Pr_C \left[\left| \frac{d(C(x), C(y))}{N} - \frac{1}{2} \right| \geq \delta \right] = \Pr_C \left[\left| Z - \frac{N}{2} \right| \geq 2\delta \cdot \frac{N}{2} \right] \leq 2 \exp\left(-4\delta^2 N/6\right) < 2^{-2n},$$

where the last inequality holds by our choice of N. ◁

By a union bound over all $x, y \in \{0,1\}^n$, Claim 16 implies the following corollary.

▶ **Corollary 17.** *Let n be a positive integer, let $\delta > 0$ and let $N = 4n/\delta^2$. Then there exists a linear code $C : \{0,1\}^n \to \{0,1\}^N$ such that for all $x \neq y \in \{0,1\}^n$,*

$$\frac{d(C(x), C(y))}{N} \in \left[\frac{1}{2} - \delta, \frac{1}{2} + \delta \right].$$

We now prove Theorem 9.

Proof of Theorem 9. Set $\delta = \sqrt{\varepsilon}/2$. Let $N = 4n/\delta^2 = 16n/\varepsilon$ and let $C : \{0,1\}^n \to \{0,1\}^{16n/\varepsilon}$ be the code obtained from Corollary 17. The following is a protocol for EQ_n.

1. Alice, on input $x \in \{0,1\}^n$ prepares state $|\phi_x\rangle := \frac{1}{\sqrt{N}} \sum_{i \in [N]} (-1)^{C(x)_i} |i\rangle$, and sends Bob $|\phi_x\rangle$.
2. Define $|\phi_y\rangle := \frac{1}{\sqrt{N}} \sum_{i \in [N]} (-1)^{C(y)_i} |i\rangle$. Bob measures with respect to the projectors $|\phi_y\rangle\langle\phi_y|$ and $I - |\phi_y\rangle\langle\phi_y|$, and outputs 1 on observing the first measurement outcome, and 0 otherwise.

This protocol succeeds with probability 1 when $x = y$. The only error arises when $x \neq y$ and Bob observes the first measurement outcome. Thus, the error probability of this protocol equals

$$\begin{aligned} \max_{x \neq y \in \{0,1\}^n} |\langle \phi_x | \phi_y \rangle|^2 &= \max_{x \neq y \in \{0,1\}^n} \left(\frac{1}{N} \sum_{i \in [N]} (-1)^{C(x)_i + C(y)_i} \right)^2 \\ &= \max_{x \neq y \in \{0,1\}^n} \left(1 - \frac{2d(C(x), C(y))}{N} \right)^2 \\ &\leq 4\delta^2 = \varepsilon, \end{aligned}$$

where the last inequality follows from Corollary 17 and the last equality follows from our choice of δ. The number of qubits sent from Alice to Bob is $\log N = \log(16n/\varepsilon) = \log(n/\varepsilon) + 4$. ◀

We show in Section 5 that the protocol in the previous proof is nearly optimal if one restricts to one-way communication with only pure states.

4.3 Quantum upper bound with mixed states

In the last section we gave a $\log(n/\varepsilon) + O(1)$ quantum upper bound on the ε-error communication complexity of EQ_n, where Alice was only allowed to send a pure state to Bob. In this section we show that allowing Alice to send a *mixed* state to Bob gives a communication upper bound that is better by a factor of 2 (which is in fact optimal). An upper bound of $\log\sqrt{n} + O(1)$ was already proved by Winter [22] for the case of constant ε, but here we obtain the correct dependence also for subconstant ε. Our protocol is based on concentration properties of overlaps of random projectors.

Consider two rank-r projectors P and Q acting on $\mathbb{C}^d$. The largest possible inner product $\mathrm{tr}(PQ)$ between them is r, which occurs iff $P = Q$. However, when one or both of the projectors are Haar-random, then we expect their inner product to be much smaller, namely only r^2/d. This is because if we take the spectral decompositions $P = \sum_{i=1}^r |u_i\rangle\langle u_i|$ and $Q = \sum_{j=1}^r |v_j\rangle\langle v_j|$, then

$$\mathrm{tr}(PQ) = \sum_{i,j=1}^{r} |\langle u_i, v_j\rangle|^2,$$

and the expected squared inner product between a random d-dimensional unit vector u_i and any fixed unit vector v_j, is $1/d$. Hayden, Leung and Winter [8, Lemma III.5] showed that this inner product is very tightly concentrated around its expectation.

▷ Claim 18 ([8, Lemma III.5]). Let P and Q be rank-r projectors on $\mathbb{C}^d$, where P is random[3] and Q is fixed. Let $\delta \in [0,1]$. Then

$$\Pr\left[\mathrm{tr}(PQ) \geq \frac{(1+\delta)r^2}{d}\right] \leq \exp\left(\frac{-r^2\delta^2}{5}\right) < 2^{-r^2\delta^2/5}.$$

The following corollary then follows by setting parameters suitably.

▶ **Corollary 19.** *For every integer $n > 0$ and all $\varepsilon \in [0, 1/2)$, there exists a set $\{P_x : x \in \{0,1\}^n\}$ of 2^n rank-r projectors on $\mathbb{C}^d$, with $r = \sqrt{10n}$ and $d = 2r/\varepsilon$, such that $\mathrm{tr}(P_x P_y) < \varepsilon r$ for all $x \neq y \in \{0,1\}^n$.*

Proof. Fix $\delta = 1$ and choose rank-r projectors $\{P_x : x \in \{0,1\}^n\}$ independently and uniformly at random. Claim 18 and our choice of parameters implies that for all $x \neq y \in \{0,1\}^n$,

$$\Pr\left[\mathrm{tr}(P_x P_y) \geq \frac{2r^2}{d}\right] = \Pr\left[\mathrm{tr}(P_x P_y) \geq \varepsilon r\right] < 2^{-r^2\delta^2/5} = 2^{-2n}.$$

The corollary now follows by applying a union bound over all distinct $x, y \in \{0,1\}^n$. ◀

We now prove Theorem 11.

Proof. Let $\{P_x : x \in \{0,1\}^n\}$ be projectors on $\mathbb{C}^d$ as guaranteed by Corollary 19, each of rank $r = \sqrt{10n}$, with $d = 2\sqrt{10n}/\varepsilon$. Our protocol for EQ_n is Protocol 2 below.

To see the correctness of this protocol, first observe that if $x = y$, then the protocol outputs the correct answer with probability 1 because $\mathrm{tr}(P_x\rho_x) = \mathrm{tr}(P_x)/r = 1$. If $x \neq y$, then the error probability is the probability of Bob observing the first measurement outcome, which is

$$\Pr[\Pi(x,y) \neq \mathsf{EQ}_n(x,y)] = \mathrm{tr}(P_y\rho_x) = \mathrm{tr}(P_y P_x)/r < \varepsilon,$$

from Corollary 19. The cost is $\log d = \log(2\sqrt{10n}/\varepsilon) \leq \log(\sqrt{n}/\varepsilon) + 3$ qubits of communication. ◀

[3] More precisely, P is a projection onto a uniformly chosen r-dimensional subspace from all r-dimensional subspaces of $\mathbb{C}^d$. We do not elaborate more on this here since it is not relevant for us.

Protocol 2 A mixed-state protocol Π for F.

1. Alice, on input $x \in \{0,1\}^n$, sends the $\log d$-qubit mixed state $\rho_x := P_x/r$ to Bob.
2. Bob, on input $y \in \{0,1\}^n$, measures w.r.t. projectors $P_y, I - P_y$, and outputs 1 on observing the first measurement outcome, and 0 otherwise.

4.4 Entanglement-assisted quantum upper bounds

We use the probabilistic method to argue the existence of a good entanglement-assisted protocol. In the following, $m \leq d$ are natural numbers to be determined later. We take the initial entangled state to be the maximally entangled state in $D = 2^d$ dimensions, i.e., d EPR-pairs:

$$|\Psi_{AB}\rangle = \frac{1}{\sqrt{D}} \sum_{i \in \{0,1\}^d} |i\rangle_A |i\rangle_B.$$

For every $z \in \{0,1\}^n$, pick independently a Haar-random element $U_z = \{|\psi_{z,r}\rangle\}_{r \in \{0,1\}^d}$ of $SU(D)$ (i.e., a random orthonormal basis is used for the 2^d columns of U_z). The following is our protocol for EQ_n:

Protocol 3 An entanglement-assisted protocol Π' for F.

1. Alice, on input $x \in \{0,1\}^n$, measures her part of $|\Psi\rangle$ in the basis U_x, obtaining $r^A \in \{0,1\}^d$. She then sends $b \equiv r_1^A r_2^A \ldots r_m^A$ to Bob (i.e., the first m bits of r^A).
2. Bob, on input $y \in \{0,1\}^n$, measures his part of $|\Psi\rangle$ in the conjugate basis of U_y, obtaining $r^B \in \{0,1\}^d$. He outputs 1 if $r_i^B = b_i$ for every $1 \leq i \leq m$, and he outputs 0 otherwise.

The one-way communication complexity of this protocol Π' is m bits. We proceed with its error analysis. After step 1, by properties of the maximally entangled state, the new joint state will be

$$|\Psi'\rangle = |\psi_{x,r^A}\rangle_A \otimes \overline{|\psi_{x,r^A}\rangle}_B$$

In particular, if $x = y$, then $r^A = r^B$ and the protocol is guaranteed to succeed. Suppose now that $x \neq y$. For $b \in \{0,1\}^m$, using the shorthand

$$R_b \equiv \{r \in \{0,1\}^d \mid r_i = b_i \ \forall i \in [m]\},$$

we find that the probability that the protocol fails (i.e., outputs 1) is given by

$$\frac{1}{D} \sum_{b \in \{0,1\}^m} \sum_{r^A, r^B \in R_b} |\langle \psi_{x,r^A} | \psi_{y,r^B} \rangle|^2.$$

Since R_b has cardinality 2^{d-m} and the expectation over the choice of U_z's of every term in the sum is 2^{-d}, we find that the expectation of the whole sum is 2^{-m}. We now only require the following concentration inequality, which is derived in [13, Chapter 3]:

▶ **Theorem 20.** *Let $F : SU(n) \to \mathbb{R}$ be a function with Lipschitz constant K with respect to the Frobenius norm,*[4] *and let μ be the uniform distribution (Haar measure) on $SU(n)$. Then, for every $\delta > 0$,*

$$\Pr_{\mu}[|F(U) - \mathbb{E}_{\mu}[F]| > \delta] < 2\exp\left(-\frac{\delta^2 n}{4K^2}\right).$$

We show:

▶ **Theorem 21.** *Let $\{\phi_r\}_{r\in\{0,1\}^d}$ be a fixed orthonormal basis of $\mathbb{C}^D$. Given $U = \{\psi_r\}_{r\in\{0,1\}^d} \in SU(D)$, define $F : SU(D) \to \mathbb{R}$ by*

$$F(U) = \sum_{b\in\{0,1\}^m} \sum_{r,r'\in R_b} |\langle\phi_r|\psi_{r'}\rangle|^2.$$

Then F has Lipschitz constant $\sqrt{D}$.

Proof. Let $U = \{\psi_r\}_{r\in\{0,1\}^d}$ and $U' = \{\psi'_r\}_{r\in\{0,1\}^d}$ be two different elements of $SU(D)$. For $b \in \{0,1\}^m$, write

$$P_b = \sum_{r\in R_b} |\phi_r\rangle\langle\phi_r|, \qquad Q_b = \sum_{r\in R_b} |\psi_r\rangle\langle\psi_r|, \qquad Q'_b = \sum_{r\in R_b} |\psi'_r\rangle\langle\psi'_r|.$$

We see that

$$F(U) = \sum_{b\in\{0,1\}^m} \operatorname{tr}(P_b Q_b) \quad \text{and} \quad F(U') = \sum_{b\in\{0,1\}^m} \operatorname{tr}(P_b Q'_b).$$

Therefore

$$\begin{aligned} F(U) - F(U') &= \sum_{b\in\{0,1\}^m} \operatorname{tr}(P_b(Q_b - Q'_b)) \le \sum_{b\in\{0,1\}^m} D_{\mathrm{tr}}(Q_b, Q'_b) \\ &\le D \sum_{r\in\{0,1\}^d} \frac{1}{D}\sqrt{1 - |\langle\psi_r|\psi'_r\rangle|^2} \\ &\le \sqrt{D^2 - \left(\sum_{r\in\{0,1\}^d} |\langle\psi_r|\psi'_r\rangle|\right)^2}. \end{aligned}$$

Here the first inequality follows from the variational characterization of trace distance ($D_{\mathrm{tr}}(Q, Q') = \max_{P:\|P\|\le 1} \operatorname{tr}(P(Q - Q'))$); the second inequality follows from the convexity of trace distance, the fact that the R_b's partition $\{0,1\}^d$, and a well-known expression for the trace distance of two pure states; and the third inequality follows from the concavity of the function $\sqrt{1 - z^2}$.

On the other hand, we can upper bound the Frobenius distance between U and U' by

$$\begin{aligned} d(U, U') &= \sqrt{\sum_{r\in\{0,1\}^d} \||\psi_r\rangle - |\psi'_r\rangle\|^2} \\ &= \sqrt{\sum_{r\in\{0,1\}^d} 2 - 2\Re(\langle\psi_r|\psi'_r\rangle)} \ge \sqrt{2D - 2\sum_{r\in\{0,1\}^d} |\langle\psi_r|\psi'_r\rangle|}, \end{aligned}$$

[4] This means $|F(U) - F(U')| \le K \cdot d(U, U')$ for all $U, U' \in SU(n)$, where $SU(n)$ is the group of $n \times n$ unitary matrices with determinant 1, the Frobenius norm $\|A\|_F$ of a matrix A is defined as $\sqrt{\sum_{i,j} |A_{ij}|^2}$, and the Frobenius distance is defined as $d(U, U') = \left\|U - U'\right\|_F$.

where the inequality uses the fact that $\Re(z) \leq |z|$ for any complex number z. We can now upper bound the Lipschitz constant of F as follows;

$$\frac{|F(U) - F(U')|}{d(U, U')} \leq \sqrt{\frac{D^2 - \left(\sum_{r\in\{0,1\}^d} |\langle\psi_r|\psi'_r\rangle|\right)^2}{2D - 2\sum_{r\in\{0,1\}^d} |\langle\psi_r|\psi'_r\rangle|}}$$
$$= \sqrt{\frac{D + \sum_{r\in\{0,1\}^d} |\langle\psi_r|\psi'_r\rangle|}{2}} \leq \sqrt{D},$$

where the last inequality is because $|\langle\psi_r|\psi'_r\rangle| \leq 1$ for each of the D r's, by Cauchy-Schwarz. ◀

For every pair of distinct inputs $x, y \in \{0,1\}^n$ and for every $\delta > 0$, it follows from the previous two results that the probability that the protocol's error probability on these inputs exceeds $2^{-m} + \delta$, is upper bounded by

$$2\exp\left(\frac{-\delta^2 D^2}{4}\right)$$

Setting $\delta = 2^{-m}$, $\varepsilon = 2^{-m+1}$ and $d = \lceil \frac{1}{2}\log_2 n + \log_2 \frac{1}{\varepsilon} + 4\rceil$, by the union bound there is a positive probability that the resulting protocol has error probability at most ε for all input pairs. This implies the existence of the desired protocol, with $m = \lceil \log 1/\delta\rceil = \lceil \log 1/\varepsilon\rceil + 1$ bits of communication.

5 Quantum one-way lower bound

In this section we prove lower bounds on the one-way quantum communication complexity of any function whose communication matrix has a large number of distinct rows. As a consequence we obtain our lower bound for EQ_n of Theorem 10.

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. We consider the model where communication is one-way, and Alice is only allowed to send a pure state to Bob. Suppose there exists a protocol of cost $\log d$ that computes F to error ε. Any such protocol looks like the following.

- Alice, on input $x \in \{0,1\}^n$, sends a message $|\phi_x\rangle$ to Bob, where $|\phi_x\rangle$ is a unit vector in $\mathbb{C}^d$.
- Bob, on input y, measures with respect to projectors $P_y, I - P_y$.

The acceptance probability of the protocol is $\|P_y|\phi_x\rangle\|^2$. Thus, we have

$$\|P_y|\phi_x\rangle\|^2 \geq 1 - \varepsilon, \quad \|(I - P_y)|\phi_x\rangle\|^2 \leq \varepsilon \quad \text{for all } x, y \in F^{-1}(1), \tag{8}$$

and

$$\|P_y|\phi_x\rangle\|^2 \leq \varepsilon, \quad \|(I - P_y)|\phi_x\rangle\|^2 \geq 1 - \varepsilon \quad \text{for all } x, y \in F^{-1}(0). \tag{9}$$

▷ Claim 22. Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function with N distinct rows in M_F. Let $X \subseteq \{0,1\}^n$ be an arbitrary subset of size N that indexes distinct rows in M_F. For a one-way quantum communication protocol as above that computes F to error $\varepsilon \leq 1/2$, we have

$$2 - 2\sqrt{\varepsilon(1-\varepsilon)} \leq \||\phi_{x_1}\rangle - |\phi_{x_2}\rangle\|^2 \leq 2 + 4\sqrt{\varepsilon}$$

for all distinct $x_1, x_2 \in X$.

Proof. Fix any two distinct $x_1, x_2 \in X$, and let $|\phi_{x_1}\rangle, |\phi_{x_2}\rangle \in \mathbb{C}^d$ be the messages sent by Alice on inputs x_1, x_2, respectively. Recall that $\||\phi_{x_1}\rangle\| = \||\phi_{x_2}\rangle\| = 1$. Because of the assumption that the rows of M_F indexed by X are all distinct, there is a $y \in \{0,1\}^n$ such that $F(x_1, y) \neq F(x_2, y)$. Without loss of generality assume $F(x_1, y) = 1$ and $F(x_2, y) = 0$. Write

$$|\phi_{x_1}\rangle = P_y|\phi_{x_1}\rangle + (I - P_y)|\phi_{x_1}\rangle,$$
$$|\phi_{x_2}\rangle = P_y|\phi_{x_2}\rangle + (I - P_y)|\phi_{x_2}\rangle.$$

Thus,

$$\begin{aligned}
\||\phi_{x_1}\rangle - |\phi_{x_2}\rangle\|^2 &= \|P_y(|\phi_{x_1}\rangle - |\phi_{x_2}\rangle)\|^2 + \|(I - P_y)(|\phi_{x_1}\rangle - |\phi_{x_2}\rangle)\|^2 \\
&\qquad \text{since } P_y \text{ and } I - P_y \text{ are orthogonal projectors} \\
&\geq (\|P_y|\phi_{x_1}\rangle\| - \|P_y|\phi_{x_2}\rangle)\|)^2 + (\|(I - P_y)|\phi_{x_1}\rangle\| - \|(I - P_y)|\phi_{x_2}\rangle\|)^2 \\
&\qquad \text{by the triangle inequality} \\
&\geq 2(\sqrt{1-\varepsilon} - \sqrt{\varepsilon})^2 \\
&\qquad \text{by Equations (8) and (9), and since } F(x_1, y) = 1 \text{ and } F(x_2, y) = 0 \\
&= 2 - 2\sqrt{\varepsilon(1-\varepsilon)}.
\end{aligned}$$

For the upper bound, first define $p := \|P_y|\phi_{x_1}\rangle\|^2 \geq 1-\varepsilon$, and $q := \|(I-P_y)|\phi_{x_2}\rangle\|^2 \geq 1-\varepsilon$.

$$\begin{aligned}
\||\phi_{x_1}\rangle - |\phi_{x_2}\rangle\|^2 &= \|P_y(|\phi_{x_1}\rangle - |\phi_{x_2}\rangle)\|^2 + \|(I - P_y)(|\phi_{x_1}\rangle - |\phi_{x_2}\rangle)\|^2 \\
&\leq (\|P_y|\phi_{x_1}\rangle\| + \|P_y|\phi_{x_2}\rangle\|)^2 + (\|(I - P_y)|\phi_{x_1}\rangle\| + \|(I - P_y)|\phi_{x_2}\rangle\|)^2 \\
&\qquad \text{by the triangle inequality} \\
&= (\sqrt{p} + \sqrt{1-q})^2 + (\sqrt{1-p} + \sqrt{q})^2 \\
&= 2 + 2\sqrt{p(1-q)} + 2\sqrt{(1-p)q} \leq 2 + 4\sqrt{\varepsilon}.
\end{aligned}$$

◁

We now state our main result of this section.

▶ **Theorem 23.** *There exists an absolute constant c such that the following holds. Let $F : \{0,1\}^n \times \{0,1\}^n$ be a Boolean function with N distinct rows in M_F. Then for all $\varepsilon \in [1/N, 1/4]$,*

$$\mathsf{Q}_\varepsilon^{\text{pure},\rightarrow}(F) \geq \log\left(\frac{\log N}{\varepsilon}\right) - \log\log\left(\frac{1}{\varepsilon}\right) - c.$$

Proof. Let $X \subseteq \{0,1\}^n$ be an arbitrary set of N elements that index distinct rows in M_F. Consider a protocol of cost $\log d$, as described in the beginning of this section, that computes F to error ε. Claim 22 implies existence of vectors $|\phi_x\rangle \in \mathbb{C}^d$ for all $x \in X$, such that

$$2 - 2\sqrt{\varepsilon(1-\varepsilon)} \leq \||\phi_{x_1}\rangle - |\phi_{x_2}\rangle\|^2 \leq 2 + 4\sqrt{\varepsilon} \tag{10}$$

for all distinct $x_1, x_2 \in X$. For each $x \in X$, define a real vector $|\phi_x^R\rangle \in \mathbb{R}^{2d}$ by

$$|\phi_x^R\rangle = \sum_{j \in [d]} |j\rangle \left(R(|\phi_x\rangle_j)|0\rangle + C(|\phi_x\rangle_j)|1\rangle\right),$$

where $R(|\phi_x\rangle_j)$ and $C(|\phi_x\rangle_j)$ denote the real and complex components of the j'th coordinate of $|\phi_x\rangle$, respectively. Note that each $|\phi_x^R\rangle$ is a unit vector, since the $|\phi_x\rangle$ are unit vectors. For all distinct $x_1, x_2 \in X$, we have

$$|\phi_{x_1}\rangle - |\phi_{x_2}\rangle = \sum_{j\in[d]} |j\rangle(R(|\phi_{x_1}\rangle_j - |\phi_{x_2}\rangle_j) + i \cdot C(|\phi_{x_1}\rangle_j - |\phi_{x_2}\rangle_j)),$$
$$|\phi^R_{x_1}\rangle - |\phi^R_{x_2}\rangle = \sum_{j\in[d]} |j\rangle((R(|\phi_{x_1}\rangle_j - |\phi_{x_2}\rangle_j)|0\rangle) + (C(|\phi_{x_1}\rangle_j - |\phi_{x_2}\rangle_j)|1\rangle)).$$

Hence, Equation (10) implies

$$\||\phi^R_{x_1}\rangle - |\phi^R_{x_2}\rangle\|^2 = \||\phi_{x_1}\rangle - |\phi_{x_2}\rangle\|^2 \in [2 - 2\sqrt{\varepsilon(1-\varepsilon)}, 2 + 4\sqrt{\varepsilon}] \tag{11}$$

for all distinct $x_1, x_2 \in X$. Since $\|v - w\|^2 = \|v\|^2 + \|w\|^2 - 2\langle v, w\rangle$ for real vectors v, w, we obtain

$$|\langle\phi^R_{x_1}|\phi^R_{x_2}\rangle| \leq 2\sqrt{\varepsilon}$$

for all distinct $x_1, x_2 \in X$. Now consider the $N \times N$ matrix M whose rows and columns are indexed by strings in X, with entries defined by

$$M_{x,y} = \langle\phi^R_x|\phi^R_y\rangle.$$

Since each $\phi^R_x \in \mathbb{R}^{2d}$, this matrix has rank at most $2d$. Since $\langle\phi^R_x|\phi^R_x\rangle = 1$ for all $x \in \{0,1\}^n$ and $|\langle\phi^R_x|\phi^R_y\rangle| \leq 2\sqrt{\varepsilon}$ for all $x \neq y \in X$, this M is a $2\sqrt{\varepsilon}$-approximation to the $N \times N$ identity matrix I. Theorem 4 implies existence of an absolute constant $c_1 > 0$ such that

$$2d \geq \mathsf{rk}(M) \geq \mathsf{rk}_{2\sqrt{\varepsilon}}(I) \geq \frac{c_1 \log N}{\varepsilon \log(1/\sqrt{\varepsilon})}.$$

Hence,

$$\log d \geq \log\left(\frac{\log N}{\varepsilon}\right) - \log\log\left(\frac{1}{\varepsilon}\right) - \log(1/c_1),$$

concluding the proof. ◀

Theorem 10 immediately follows from Theorem 23 since all 2^n rows in M_{EQ_n} are distinct.

6 Future work

We mention some possible directions for future work:

- Those of our lower bounds that use Alon's approximate-rank bound (Theorem 4) lose an additive $\log\log(1/\varepsilon)$. This term is necessary in some regimes, in particular when ε is very small ($\sim 2^{-n}$) and n/ε gets bigger than the trivial dimension upper bound 2^n. However, in some regimes it may be avoidable. Also Alon's bound itself might be slightly improvable.
- We leave open the optimal quantum communication complexity of Equality with small error in the *simultaneous message passing* (SMP) model, where Alice and Bob each send a message to a "referee" who has to decide the output. With public randomness $\log(1/\varepsilon) \pm O(1)$ classical bits of communication are necessary and sufficient, but with private randomness it is not clear. In the classical case, $\Theta(\sqrt{n})$ bits of communication are necessary [17] and sufficient [2] for constant error. In the quantum case, $\Theta(\log n)$ qubits are necessary and sufficient [4] for constant error. One can get an $O(\log(n)\log(1/\varepsilon))$ ε-error upper bound by repeating the quantum fingerprinting protocol of Buhrman et

al. [4] $O(\log(1/\varepsilon))$ times, but that is much worse than the $\log(\sqrt{n}/\varepsilon)$ and $\log(n/\varepsilon)$ upper bounds that we have in the one-way mixed-state and pure-state scenarios (Theorems 11 and 9). In neither the randomized nor the quantum SMP settings do we have tight bounds for small ε.

- We also leave open the optimal communication complexity of equality with small error in the entanglement-assisted setting. The classical public-coin protocol and the one we exhibited both require $\lceil \log(1/\varepsilon) \rceil + O(1)$ bits of communication to compute EQ_n to within error ε, and it seems probable that this is essentially optimal.

References

1. Noga Alon. Perturbed identity matrices have high rank: Proof and applications. *Combinatorics, Probability and Computing*, 18(1-2):3–15, 2009. doi:10.1017/S0963548307008917.
2. Andris Ambainis. Communication complexity in a 3-computer model. *Algorithmica*, 16(3):298–301, 1996.
3. Anurag Anshu, Naresh Goud Boddu, and Dave Touchette. Quantum log-approximate-rank conjecture is also false. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 982–994, 2019. doi:10.1109/FOCS.2019.00063.
4. Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16), September 26, 2001. arXiv:quant-ph/0102001.
5. Harry Buhrman and Ronald de Wolf. Communication complexity lower bounds by polynomials. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity (CCC)*, pages 120–130, 2001. doi:10.1109/CCC.2001.933879.
6. Arkadev Chattopadhyay, Nikhil S. Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. *Journal of the ACM*, 67(4):23:1–23:28, 2020. Earlier version in STOC, 2019. URL: https://dl.acm.org/doi/10.1145/3396695.
7. Hamza Fawzi, João Gouveia, Pablo A. Parrilo, Richard Z. Robinson, and Rekha R. Thomas. Positive semidefinite rank. *Mathematical Programming*, 153(1):133–177, 2015. doi:10.1007/s10107-015-0922-1.
8. Patrick Hayden, Debbie W. Leung, and Andreas Winter. Aspects of generic entanglement. *Communications in Mathematical Physics*, 265(1):95–117, 2006.
9. Matthias Krause. Geometric arguments yield better bounds for threshold circuits and distributed computing. *Theoretical Computer Science*, 156(1&2):99–117, 1996. doi:10.1016/0304-3975(95)00005-4.
10. Ilan Kremer. Quantum communication. Master's thesis, Hebrew University, Computer Science Department, 1995.
11. Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.
12. Troy Lee, Zhaohui Wei, and Ronald de Wolf. Some upper and lower bounds on psd-rank. *Mathematical Programming, Series A*, 162(1–2):495–521, 2017.
13. Elizabeth Meckes. Concentration of measure and the compact classical matrix groups, 2014. Lecture notes, available at https://case.edu/artsci/math/esmeckes/Haar_notes.pdf.
14. Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. doi:10.1017/CBO9780511813603.
15. Ashwin Nayak. *Lower bounds for Quantum Computation and Communication*. PhD thesis, University of California, Berkeley, 1999.
16. Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991. doi:10.1016/0020-0190(91)90157-D.
17. Ilan Newman and Mario Szegedy. Public vs. private coin flips in one round communication games. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 561–570, 1996.

18 Michael A. Nielsen. *Quantum Information Theory*. PhD thesis, University of New Mexico, Albuquerque, 1998.
19 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
20 Anup Rao and Amir Yehudayoff. *Communication Complexity and Applications*. Cambridge University Press, 2020.
21 Makrand Sinha and Ronald de Wolf. Exponential separation between quantum communication and logarithm of approximate rank. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 966–981, 2019. doi:10.1109/FOCS.2019.00062.
22 Andreas Winter. Quantum and classical message identification via quantum channels. In O. Hirota, editor, *Festschrift A.S. Holevo 60*. Rinton, 2004. arXiv:quant-ph/0401060.
23 Ronald de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–353, 2002. doi:10.1016/S0304-3975(02)00377-8.
24 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.
25 Andrew Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of the 34th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 352–361, 1993.

A Quantum communication complexity and psd-rank

In this section, we prove Theorem 3, restated below.

▶ **Theorem 24** (Restatement of Theorem 3). *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $\varepsilon > 0$. Then,*

$$\mathsf{Q}^{\mathsf{pri}}_{\varepsilon}(F) \geq \log \mathsf{rk}^{\mathrm{psd}}_{\varepsilon}(M_F) + 1.$$

Proof. Consider an ℓ-qubit protocol for F, without public randomness. Because private randomness can be generated using Hadamard gates, we will assume the protocol is unitary, with only a measurement of the output qubit at the end. Let the starting state of the protocol be $|x0^s\rangle_A|y0^s\rangle_B|0\rangle_C$, where the first and second parts are Alice and Bob's register, respectively (containing their input and s workspace qubits each), and the third part is the channel qubit. It is easy to prove by induction that after ℓ qubits of communication, the final state of a protocol has the following form (first observed by Kremer [10] and Yao [25]):

$$\sum_{i\in\{0,1\}^\ell} |a_i(x)\rangle|b_i(y)\rangle|i_\ell\rangle,$$

where $|a_i(x)\rangle, |b_i(y)\rangle$ are subnormalized quantum states. Let P denote the acceptance probability matrix, i.e., $P(x,y)$ is the probability that the protocol outputs 1 on input (x,y). We assume without loss of generality that the output qubit is the last qubit put on the channel. We have

$$P(x,y) = \left\| \sum_{i\in\{0,1\}^\ell : i_\ell = 1} |a_i(x)\rangle|b_i(y)\rangle|i_\ell\rangle \right\|^2 = \sum_{i,i'\in\{0,1\}^\ell : i_\ell = i'_\ell = 1} \langle a_i(x)|a_{i'}(x)\rangle \cdot \langle b_i(y)|b_{i'}(y)\rangle.$$

For each $x \in \{0,1\}^n$ define a $2^{\ell-1} \times 2^{\ell-1}$ matrix A_x with rows and columns indexed by strings $i, i' \in \{0,1\}^{\ell-1} \times \{1\}$:

$$A_x(i,i') = \langle a_i(x)|a_{i'}(x)\rangle.$$

Similarly, for each $y \in \{0,1\}^n$ define a $2^{\ell-1} \times 2^{\ell-1}$ matrix B_y by

$$B_y(j,j') = \langle b_j(y) | b_{j'}(y) \rangle.$$

These A_x and B_y are Gram matrices and hence psd. Moreover it is easy to verify that $P(x,y) = \mathrm{tr}(A_x B_y)$. Since the protocol makes error at most ε on each input, the matrix P entrywise approximates M_F up to ε. Hence $\mathsf{rk}^{\mathrm{psd}}_{\varepsilon}(M_F) \leq 2^{\ell-1}$. Taking logarithms gives the theorem. ◀

B Approximate-rank upper bounds for distributed SINK function

In this section we show improved upper bounds on the approximate nonnegative-rank and approximate psd-rank of $M_{\mathsf{SINK} \circ \mathsf{XOR}}$, where SINK is defined as follows.

▶ **Definition 25.** *Define the function* $\mathsf{SINK}_n : \{0,1\}^n \to \{0,1\}$ *on* $n = \binom{m}{2}$ *inputs as follows. The inputs are viewed as orientations of edges on a complete graph with* m *vertices. The function outputs 1 if there is a sink in the graph, and 0 otherwise.*

Consider the function $\mathsf{SINK}_n \circ \mathsf{XOR} : \{0,1\}^{2n} \to \{0,1\}$. This function was recently used to refute the randomized and quantum versions of the log-rank conjecture [6, 21, 3]. Chattopadhyay, Mande and Sherif [6, Theorem 1.10] showed that the 1/3-approximate rank of $M_{\mathsf{SINK}_n \circ \mathsf{XOR}}$ is $O(m^4)$ and the 1/3-approximate nonnegative-rank of $M_{\mathsf{SINK}_n \circ \mathsf{XOR}}$ is $O(m^5)$. As a consequence of our improved upper bounds for the ε-approximate nonnegative-rank of the Identity matrix (Corollary 8), we are able to use the same proof idea as theirs to obtain an $O(m^4)$ upper bound on the 1/3-approximate nonnegative-rank of $M_{\mathsf{SINK}_n \circ \mathsf{XOR}}$, matching the approximate rank upper bound. We also obtain approximate psd-rank upper bounds for $\mathsf{SINK}_n \circ \mathsf{XOR}$.

▷ Claim 26. Let m be a positive integer, let $n = \binom{m}{2}$. Then,

$$\mathsf{rk}^{+}_{1/3}(M_{\mathsf{SINK}_n \circ \mathsf{XOR}}) = O(m^4)$$
$$\mathsf{rk}^{\mathrm{psd}}_{1/3}(M_{\mathsf{SINK}_n \circ \mathsf{XOR}}) = O(m^{2.5}).$$

Proof. Note that $\mathsf{SINK}_n \circ \mathsf{XOR}$ can be expressed as a *sum* of m Equalities, each with $2(m-1)$ inputs, one corresponding to each vertex in the underlying graph for SINK. Recall that the communication matrix of Equality is the Identity matrix. We require sub-additivity of nonnegative-rank and psd-rank, which are both easy to verify.

- Corollary 8 implies that each of these Equalities have $(1/3m)$-approximate nonnegative-rank $O(m^3)$. Summing up these m matrices, we conclude that the (1/3)-approximate nonnegative-rank of $\mathsf{SINK}_n \circ \mathsf{XOR}$ equals $O(m^4)$.
- Corollary 12 implies that each of these Equalities have $(1/3m)$-approximate psd-rank $O(m^{1.5})$. Summing up these m matrices, we conclude that the (1/3)-approximate psd-rank of $\mathsf{SINK}_n \circ \mathsf{XOR}$ equals $O(m^{2.5})$. ◁

Revisiting Mulmuley: Simple Proof That Maxflow Is Not in the Algebraic Version of NC

Ulysse Léchine
LIPN, Paris, France
IRIF, Paris, France

Abstract

We give an alternate and simpler proof of the fact that PRAM without bit operations (shortened to iPRAM for integer PRAM) as considered in paper [2] cannot solve the maxflow problem. To do so we consider the model of PRAM working over real number (rPRAM) which is at least as expressive as the iPRAM models when considering integer inputs. We then show that the rPRAM model is as expressive as the algebraic version of NC : algebraic circuits of fan-in 2 and of polylog depth noted $\mathsf{NC}^{\mathsf{alg}}$. We go on to show limitations of the $\mathsf{NC}^{\mathsf{alg}}$ model using basic facts from real analysis : those circuits compute low degree piece wise polynomials. Then, using known results we show that the maxflow function is not a low-degree piece-wise polynomial. Finally we argue that $\mathsf{NC}^{\mathsf{alg}}$ is actually a really limited class which limits our hope of extending our results to the boolean version of NC.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic complexity, P vs NC, algebraic NC, GCT program

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.33

1 Introduction

1.1 Previous work

An important question in complexity theory is determining if $P = NC$, where P is the set of languages solvable by a polynomial time Turing machine and NC the set of languages solvable by uniform family of poly-logarithmic depth, polynomial size circuit. It is thought that $P \neq NC$ but no one has come close to proving it. The maxflow problem is a problem in P where the inputs are directed graphs whose edges are labelled with integers called capacities, a specified node s called the source, a specified node t called the sink, and an integer threshold k, the input is in the language if a flow greater than k units may flow from s to t. The maxflow problem is known to be P complete with regards to NC reductions, meaning that membership of the maxflow problem to NC is equivalent to $P = NC$. In his celebrated result [2], Mulmuley defines a computation model called integer PRAMs (iPRAM) without bit operation which is akin to the PRAM model (communicating processors working in parallel, see [2] for a formal definition) except that inputs are integers (instead of bits) and the usual bit operations are replaced by multiplication, addition and comparison (all of those take $\Omega(1)$ time regardless of the size of the integers). He then proves that the maxflow problem cannot be computed by iPRAM with polynomially many processors running in polylogarithmic time. The model of iPRAM with polynomially many processors running in polylogarithmic time is basically an algebraic version of NC, which we denote by $\mathsf{NC}^{\mathsf{alg}}$, where the inputs are now integers and the gates multiplication, addition and comparison, and the fan-in of each gate is at most 2.[1] Moreover it was also known that the maxflow problem does have algebraic algorithm (working only using multiplication and addition)

[1] making the fan-in to be arbitrary wouldn't change the model, we can replace a large fan-in gate with a low height tree of gates

© Ulysse Léchine;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 33; pp. 33:1–33:12
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

running in polynomial time, so in short Mulmuley proved that the maxflow problem is not in the algebraic version of NC yet it is in the algebraic version of P. So in the algebraic world $NC \neq P$.

Size of inputs

For now we have very loosely spoken of "polynomial time", "poly-logarithmic depth"... it is to be understood that we mean "polynomial time in the size of the input", but what is the size of an input? It may seem trivial, but we insist upon it to clarify any future misconceptions. In the bit world (where languages are subsets of $\{0,1\}^*$, and inputs bitstrings), the size of the input is its number of bits. In the algebraic world (where languages can be thought of as subsets of $\mathbb{N}^*$ and inputs strings of numbers) the size of an input is its number of numbers. For instance let us consider a specific instance of a maxflow problem, say a weighted directed graph G with 10 nodes given via its matrix representation, two integers to specify which node is the source and which is the sink, and a integer threshold k. In the algebraic world the size of this instance is $10^2 + 2 + 1 = 103$. In the bit world we do not have enough information to conclude because we do not know the bitlength of the capactities. This is just to shed light on the difference between the two models, in the algebraic one only the number of integers count, no matter their bitlength. Now when showing impossibility results, for instance that maxflow is not in the algebraic version of NC, we will on top of it prove that the impossibility still stand even if we restricts integers to be of bitlength polynomial in the size of the input (i.e. polynomial in the number of numbers). This is interesting because it "levels the playing field" between the algebraic world and the bit world, if the algebraic world comes short of solving the maxflow problem it will not be because of inputs of very large bitlengths[2].

1.2 Our result

In this paper we show that maxflow is not in the algebraic version of NC $\mathsf{NC}^{\mathsf{alg}}$ (subsequently also not in iPRAM). We detail things as follows: our technique for showing the result. Contributions and differences between our paper and Mulmuley's. Discussion on the meaning of this result.

1.2.1 Technique

1.2.1.1 The different PRAM models and algebraic circuits are equivalent

Firstly we consider (in order to use real analysis) real PRAM noted rPRAM (introduced in [3]). rPRAM are akin to iPRAM except inputs are real numbers instead of integers, on which we can perform addition, multiplication, *division* and comparison[3]. We argue efficiently that rPRAM are at least as potent as iPRAM when we restrict the inputs to integers (iPRAM are not defined otherwise). Then we consider the non-uniform[4] family of algebraic circuits

[2] This is how size was handled in [2] hence our choice. Another way of defining the size of inputs in the algebraic world, perhaps more intuitive or familiar to some readers, is to include the bitlength of the inputs in the size. As a side note changing size like this would strengthen the class $\mathsf{NC}^{\mathsf{alg}}$ and our proofs would still work.

[3] the division is probably superfluous when considering the decision version of a problem (we might be able to simulate it using multiplication and addition) but it makes our model more potent in the general case. In any case if the goal is only to prove the original result of Mulmuley, one may (on second reading) disregard the division operation in the rest of the paper, it would make proofs even simpler

[4] uniformity never plays a role so we may as well not restrict our circuits

whose inputs are (multiple) real numbers, whose gates are multiplication, addition, division, comparison (all of fan-in 2), whose depth is poly-logarithmic in the size of the input and whose largeness is polynomial in the size of the input. We call this class $\mathsf{NC}^{\mathsf{alg}}$, it is to be thought of as the algebraic equivalent of NC. We then give an argument to show that under a reasonable time slowdown and largeness blow up, $\mathsf{NC}^{\mathsf{alg}}$ can simulate rPRAM with polynomially many processors running in poly-logarithmic time. Now we can focus our attention on proving lowerbounds for $\mathsf{NC}^{\mathsf{alg}}$ which is much simpler than with rPRAM or iPRAM (but any lower bounds for $\mathsf{NC}^{\mathsf{alg}}$ entail ones for rPRAM and iPRAM). Considering $\mathsf{NC}^{\mathsf{alg}}$ instead of iPRAM is one of the key ingredient to the simplicity of our proof, the other one being restricting our attention to algebraic circuit over one variable.

1.2.1.2 $\mathsf{NC}^{\mathsf{alg}}$ are really limited

First we prove by induction that a family of algebraic circuits whose input is only one number are really simple functions: they are piece-wise polynomials [5] and each of those polynomials is of small degree (we bound the degree and the number of different polynomials by the depth of the circuit, see 2). So for instance any function $f : \mathbb{R} \mapsto \mathbb{R}$ with too many slope changes cannot be computed by an algebraic circuit of small depth.

1.2.1.3 Problem

Then we explain how one can use this result over one input algebraic circuits to limit the complexity of algebraic circuits over multiple inputs: let us say we want to prove that a family of functions $g_n : \mathbb{R}^n \mapsto \mathbb{R}$ is not computable in $\mathsf{NC}^{\mathsf{alg}}$ (i.e. with depth polylog(n)) then one just needs to find a family of functions $f_n : \mathbb{R} \mapsto \mathbb{R}^n$ computable with a circuit of depth polylog(n) such that $g_n \circ f_n$ (which is now a function from $\mathbb{R}$ to $\mathbb{R}$ so we can use our theorem) is complex enough not to be computed by algebraic circuits of depth polylog(n). Since f_n is computable by such circuits then it must be g_n which is not. Now for the maxflow problem specifically: call $g_n : \mathbb{R}^{n^2} \mapsto \mathbb{R}$ the function which takes a weighted directed graph given as its matrix representation and returns its maximum flow (the source is node 1 and the sink node n). One simply needs to find a simple function $f_n : \mathbb{R} \mapsto \mathbb{R}^n$ such that $g_n \circ f_n$ is too complex to be computed by circuits of depth polylog(n). Such a family of functions f_n has been know for a long time ([1], Mulmuley even gives his own in [2]), and they are actually very simple since they are affine functions, so definitely computable by algebraic circuits even of depth 1. Then we simply conclude that g_n in not computable by circuits of $\mathsf{NC}^{\mathsf{alg}}$.

1.2.2 Contributions and differences

The contributions of this paper are twofold. Firstly the simplicity of the proof when compared with Mulmuley's. We use very basic (univariate) polynomial analysis to limit the complexity of algebraic circuits, and then we use a known result about the parametric complexity of the maxflow problem to immediately conclude. Thanks to our proof this quite famous result may be proven to researchers or student in an hour or so. Secondly, because of the simplicity of the proof we feel it is much easier to understand the true limitations of $\mathsf{NC}^{\mathsf{alg}}$ (therefore iPRAM) and reveal that they are basically humongous and why even basic functions cannot be computed in this model of computation. Mulmuley knew about these limitations but we are more pessimistic than him. Additionally our technique exposes the fundamental and

[5] actually polynomial fractions but this is an unimportant detail

non-trivial link between computing on inputs of large bitlength and computing functions on many inputs of smaller bitlengths : we show that functions over one input cannot be computed even if we restrict the inputs to $x \in [0, 2^{n^d}]$ (where d is an integer) and we use it to conclude that some function g over $e \in \mathbb{N}$ inputs cannot be computed even if we restrict its domain to $[0, 2^n]^e$. The technique that Mulmuley uses is based on much more complicated mathematics and with a model (iPRAM) on which proofs are clumsier that on circuits. He basically uses the same plan and gives the same argument for the non-membership of maxflow to iPRAM: iPRAM may only compute "simple" multivariate-polynomial-like functions but using parametrization we can show that maxflow is not a simple function. To be clear, most of our mathematical arguments may be - from a logically formal standpoint - already contained in Mulmuley's but we do not think it makes this paper redundant at all. Those arguments were not readily apparent in Mulmuley. It is only after careful examination that one realizes that our mathematical arguments are embedded in the ones of Mulmuley's. Our proof, because we limit the expressiveness of univariate circuits to polynomial of low degree, very clearly exposes the shotcomings of the expressivity of the algebraic model. Those shortcomings were was not as apparent in Mulmuley's paper, because he deals with harder concepts (more or less multivariate polynomials): limitations on such objects do not seem as restrictive, and it still seems that we can compute "interesting" functions in iPRAM, so it gives a false sense of how expressive the class iPRAM is. On the other hand the approach used by Mulmuley, although it lacks in simplicity, may allow one to add operations on top of addition and multiplication, for instance he claims his results are still valid if one allows computing the parity of an integer. It is not immediately clear that our proof techniques can be salvaged if we add such an operation (adding the parity function allows us to create a polynomial with exponentially many pieces). Also Mulmuley shows that randomness still does not allow us to compute the maxflow problem. We do not consider randomness in this paper, our educated guess is that our technique can be fitted to include randomness. One last hiccup is that for what seems could be purely technical reasons we were not able to show that the *decision* version of the maxflow problem cannot be computed in $\mathsf{NC}^{\mathsf{alg}}$. We only show that the search version (taking a graph and outputting its maximal flow) is not in $\mathsf{NC}^{\mathsf{alg}}$. At the end of the paper we give promising leads to solve this slight issue.

1.2.3 Discussion

As explained before we believe our results shed light on the lack of potency of the models considered by Mulmuley and consequently ours. We are still far from proving that $NC \neq P$. To cement that idea we show that trivial problem for bit complexity class, like checking the parity of a bitstring (its last bit is 0) is obviously in NC (its even in AC^0), but it is not in $\mathsf{NC}^{\mathsf{alg}}$. This was known of Mulmuley but he somewhat disregards it, we argue later in the paper that it should not be disregarded. We, very humbly, think it somewhat dampens the hope of using such techniques to prove $P \neq NC$, or at the very least that closing the gap using such techniques will be quite involved. Secondly we warn the reader: although the maxflow is special in the bit complexity class P, as it is complete, it is as far as we can tell not special in the algebraic world: we could not find general reductions from problems in algebraic-P to the algebraic maxflow problem. The maxflow problem is only special in the sense as it has an intuitive translation from the bit world to the algebraic world: every capacity gets mapped to one number. If we had taken another problem like primality testing for instance, the translation would not make much sense. Indeed in primality testing we have a number x and must determine if it is prime, its intuitive translation in the algebraic world would be similar: we have an integer x and must determine if it is prime. The issue is that in the algebraic world our input size would now be 1 (we have 1 number) and so circuits should now be of constant size.

All of that being said it is still interesting in its own right to know that the maxflow problem cannot be solved in poly-logarithmic time by polynomially many processors. Indeed algorithms used to solve the maxflow problem are, to our knowledge, all algebraic, they do not use the bitwise representation of numbers, only their absolute value. So it gives credible reason to believe $P \neq NC$

2 Equivalence between PRAMS and algebraic circuits

We will start with definitions of classes iPRAM, rPRAM and $\mathsf{NC}^{\mathsf{alg}}$. Our definition of iPRAM is taken almost verbatim from Mulmuley's paper [2]. Our definition of rPRAM is taken from [3]

▶ **Definition 1** (iPRAM). *It is like the usual PRAM model, the main difference being that its input are integers (instead of bits) and it does not provide instructions for any bit operations such as $\wedge$, $\vee$, or extract-bit. The model provides usual arithmetic ($+$, $-$, $\times$), comparison ($=$, $\leq$, $<$), store, indirect reference, and branch operations. Each memory location can hold an integer; a rational number is represented by a pair of integers – its numerator and denominator – both of which can be accessed by the processors separately. The processors communicate as in the usual PRAM model. The model does not provide instructions for truncation or integer division with rounding. The lower bounds are proved in the conservative unit-cost model in which each operation is assigned unit cost regardless of the bitlengths of the operands.*

In Mulmuley's setting indirect addressing is not allowed to depend in the input, for instance we may not use a capacity as an address. There need not be such restrictions in our model.

▶ **Definition 2** (rPRAM). *It is like the* iPRAM *model except the inputs are real numbers and we have an additional / (divide) gate. Furthermore each value is now stored in a memory location represented by a real number, i.e. we have an instruction* <u>*store value x in cell y*</u> *where both x and y are real numbers. We may also retrieve the content of a memory cell indexed by y a real number.*

Notice that even though our memory cells are labelled by real numbers, at most polynomially many labels can be instantiated over the course of a computation using polynomially many processors and running in poly-logarithmic time.

In order to meaningfully compare the two models (iPRAM and rPRAM) we may restrict our view to integer inputs, in this setting it is clear enough that any computation made by an iPRAM can be made by an rPRAM.

▶ **Definition 3** (Algebraic circuit). *An algebraic circuit is a circuit with gates $+, \times, -, /, =, <, \leq$ of fan-in 2, and constant gates n for all $n \in \mathbb{N}$. Its inputs are real numbers. The comparison gates $=, <, \leq$ return 1 when true, 0 otherwise. All the other gates are interpreted in the obvious way.*

▶ **Definition 4** (log-depth algebraic circuits). *C_n is a sequence of* $\mathsf{NC}^{\mathsf{alg}}$ *(polylog-depth algebraic circuits) if each C_n is an algebraic circuit working over n inputs, the size of each circuits grows polynomial in n and the depth polylogarithmically in n.*

We restrict the largeness of log-depth algebraic circuits to polynomials. If unrestricted they could be bigger, indeed even if we work with fan in 2 a circuit of depth $\log^d(n)$ could be of largeness as big as $2^{\log^d(n)}$ which is bigger than any polynomial for $d > 1$. Mulmuley

considers such circuits in his paper, for simplicity we do not although our results would still stand. We now sketch a proof that $\mathsf{NC}^{\mathsf{alg}}$ (circuits) and $rPRAM$ (prams) with polynomially many processors and of running time polylogarithmic define the same class of functions.

▶ **Theorem 1.** $\mathsf{NC}^{\mathsf{alg}}$ *(circuits) and $rPRAM$ (prams) with polynomially many processors and of running time polylogarithmic define the same class of functions.*

Proof. It is not hard to see that one can simulate a circuit using a real PRAM. Just have one processor per gate.

The other direction is not so trivial, here is a sketch of the proof. To handle instructions not related to memory manipulation we may use techniques used in the traditional Turing-machine-to-circuit conversions, the tricky part is taking care of the memory addressing using real numbers. Proof of this fact when considering NC and PRAMs (in the boolean setting) can be found in introductory computer science books. Let us consider p processors in parallel running for k steps. Our circuit will consist of k layers $(l_i)_{i=1}^k$, each layers contains pk gates representing the memory state of each processor (each processor modifies at most k internal addresses over the whole run), plus an additional number of gates representing the state of the shared memory. In order to deal with the processor accessing and modifying the shared memory we need to be careful, indeed the index of the cell accessed by a processor is unknown across a range of infinitely many indices, to deal with this issue we use a trick known as dynamic addressing, each memory cell will be represented by a pair of gates one containing the actual content and one containing the index of the cell. Only kp memory cells can be used at most by the processors so we add to each layer a $2kp$ gates representing the state of the share memory (index and content). Moreover all operations on the memory (like checking if a certain memory cell given by its address is empty) can be simulated using a overhead of $O(\log(kp))$. In total our circuit is of size polynomial in pk and of depth $O(k\log(pk))^2$ (the square comes from the Turing machine to circuit conversion). ◀

3 Limitations of log-depth algebraic circuits

In this section we give a direct proof that log-depth algebraic circuits over one variable compute piece-wise polynomial fractions of low degree and few pieces (namely less than 2^n if the depth is polylog(n)). This is detailed in main theorem 2.

▶ **Definition 5** (Interval). *A set I over $\mathbb{R}$ is an interval if there exists $a, b \in (\mathbb{R} \cup \{-\infty; +\infty\})^2$ two real numbers such that $I = [a, b]$ or $I =]a, b]$ or $I = [a, b[$ or $I =]a, b[$.*[6]

▶ **Definition 6** (Comparison of intervals). *Let I and J be two non-empty, non-intersecting intervals over $\mathbb{R}$. We write $I < J$ if $\forall x \in I, \forall y \in J, x < y$. Let us have a collection of r intervals $(I_i)_{i=0}^r$, we write $I_1 < I_2 < \ldots < I_r$ to indicate that they are given in order and they are all non-empty and non-intersecting.*

▶ **Definition 7** (Interval cut and pieces). *Given a family $(I_i)_{1 \leq i \leq r}$ of r intervals, $I_1 < I_2 < \ldots < I_r$ we say it is an interval cut of size r if $I_1, I_2, \ldots, I_r$ partitions $\mathbb{R}$. Each I_i is called a piece.*

In the following we may use as a shorthand the notation I for a family of r intervals $(I_i)_{1 \leq i \leq r}$.

[6] since some reader may no be familiar with these notations: $]a, b]$ is equal to $[a, b] \setminus a$

▶ **Definition 8** (Intertwining interval cuts). *Given an interval cut of size n $I_1 < I_2 < \ldots < I_n$ and an interval cut of size m $J_1 < J_2 < \ldots < J_m$. There is at least one interval cut of k-pieces $K_1, \ldots, K_k$ such that $\forall h \in [k], \exists i, j, K_h \subset I_i \wedge K_h \subset J_j$, if k is minimal then we call $K_1, \ldots, K_k$ an intertwining of I and J.*

Note in the previous definition that when k is minimal it is smaller than $n + m$ (this is not hard to see). As an example of what is an intertwining consider the family $I = (]-\infty; 0],]0; +\infty[)$ and $J = (]-\infty; 1],]1; +\infty[)$, then one possible value of an intertwining is $(]-\infty; 0],]0; 1],]1; +\infty[)$.

▶ **Definition 9** (Piece-wise polynomial fraction). *A function f over $D \subset \mathbb{R}$ is said to be a piece-wise polynomial fraction if there exists $n \in \mathbb{N} \setminus \{0\}$ and an interval cut $I_1 < I_2 < \ldots < I_n$ and n polynomial fractions $f_1, f_2, \ldots, f_n$ such that $\forall i \in [\![1; n]\!], \forall x \in I_i \cap D, f(x) = f_i(x)$.* [7]

Given such an interval cut $I_1 < I_2 < \ldots < I_n$ we may say f is a piece-wise polynomial fraction over I. The number of pieces of a piece-wise polynomial fraction is the smallest number n such that there exist $I_1 < I_2 < \ldots < I_n$ as described above. Let us write c the function which takes a piece-wise polynomial fraction f and return its number of pieces (we extend c to also take circuits as inputs when they compute piece-wise polynomial fractions).

For clarification the poles of a polynomial fraction do not add new pieces, for instance the function $f(x) = \frac{1}{x^2-1}$ defined for all $x \in \mathbb{R} \setminus \{-1; 1\}$ is a one piece polynomial fraction. If one wanted to have functions defined over the whole of $\mathbb{R}$ it would be no issue as whenever performing a division we could add a comparison gate in the circuit to verify that we are not dividing by 0, and if so return an arbitrary value.

▶ **Definition 10** (Augmented degree of a polynomial fraction). *Let $F = \frac{P}{Q}$ be a polynomial fraction over $\mathbb{R}$, meaning that P and Q are coprime polynomials over $\mathbb{R}$, then we define its augmented degree noted $d(F)$ as $d(F) = d(P) + d(Q) + 1$ (where $d(P)$ and $d(Q)$ denote the usual degree of polynomials)*

▶ **Definition 11** (Augmented degree of a piece-wise polynomial fraction). *Let f be a piece-wise polynomial fraction over $\mathbb{R}$ with a minimal interval cut $I_1 < \ldots < I_n$ and $f_1, \ldots, f_n$ the associated polynomial fractions then we define $d(f) = max_{0 \leq i \leq n} d(f_i)$.*
It is routine to check that this definition does not depend on the chosen interval cut.

This is the main theorem of the paper on which we derive all impossibility results of section 4.

▶ **Theorem 2.** *A circuit P of depth k taking only one input, computes a function which is a piece-wise polynomial fraction. Let us call with $c(P)$ its number of pieces and $d(P)$ its augmented degree, then $d(P) \leq 2^k$ and $c(P) \leq 2^{\frac{k^2+3k}{2}}$*

Proof. We prove the result for each gate by induction on the depth of the gate, both that we do indeed compute a piece-wise polynomial fraction and that the bounds are correct. Let us denote c_k the maximal number of pieces of a function computed by a gate at depth k. Let us denote d_k the maximal augmented degree of a function computed by a gate at depth k. We are going to prove that $d_k \leq 2^k$ and $c_k \leq 2^{\frac{k^2+3k}{2}}$ by induction.

For depth $k = 1$ the function computed is either $f = (x \mapsto x)$ or $g = (x \mapsto 1)$. Notice that $d(f) = 2$, $c(f) = 1$, $d(g) = 1$ and $c(g) = 1$. Therefore $d_1 \leq 2^1$ and $c_1 \leq 2^2$.

[7] $[\![1; n]\!]$ denotes the integers in the interval $[1; n]$

Going from k to $k+1$: Let us consider a gate called p at depth $k+1$. Our induction hypothesis is that $d_k \leq 2^k$ and $c_k \leq 2^{\frac{k(k+1)}{2}}$.

Multiplication gate: $p = p_1 * p_2$. Let $r = c(p_1)$ and $m = c(p_2)$ we consider the interval cuts of p_1 and p_2 which we denote $I_1 < \ldots < I_r$ and $J_1, < \ldots < J_m$, let us consider $K_1, \ldots, K_k$ an intertwining of I and J such that $k < r + m$, note that p is a piece-wise polynomial fractions over K. Therefore $c(p) \leq c(p_1) + c(p_2) \leq 2 * c_k \Rightarrow c_(p) \leq 2 * c_k \leq 2^{\frac{(k+1)(k+2)}{2}}$. By reasoning over each piece of the interval cut K we have that $d(p) \leq d(p_1) + d(p_2) \leq 2d_k$.

Addition and division gate work in a similar fashion.

Comparison gate: $p = (p_1 \leq p_2)$. Let us define $n = c(p_1)$ and $m = c(p_2)$ and let $A_0 \leq A_1 \leq \ldots \leq A_n$ be the pieces of p_1, $B_0 \leq B_1 \leq \ldots \leq B_m$ the pieces of p_2. Let $I_0 \leq I_1 \leq \ldots \leq I_r$ be an intertwining of A and B. For all i, p_1 and p_2 are both polynomial fraction over I_i and we can canonically write $p_{i,1} = \frac{q_{i,1}}{t_{i,1}}$ and $p_{i,2} = \frac{q_{i,2}}{t_{i,2}}$. We want to count the number of pieces of p, to that end we focus on each I_i one by one. Notice that over I_i any new piece of p contained in I_i may only occur on places where $p_{i,1} - p_{i,2}$ changes sign, moreover $p_{i,1} = p_{i,2} \Leftrightarrow q_{i,1} * t_{i,2} - t_{i,1} * q_{i,1} = 0$. But the polynomial $q_{i,1} * t_{i,2} - t_{i,1} * q_{i,1}$ has at most $d(p_1) + d(p_2) - 1$ roots, therefore over I_i we "add" at most $d(p_1) + d(p_2)$ pieces. Since we have at most $r \leq n + m = c(p_1) + c(p_2)$ pieces I_i, we have that c_p is at most $(d(p_1) + d(p_2)) * (c(p_1) + c(p_2)) \leq 2d_k * 2c_k \leq 2 * 2^k * 2 * 2^{\frac{k^2+3k}{2}} \leq 2^{\frac{(k+1)^2+3(k+1)}{2}}$. And $d(p)$ is equal to $1 \leq 2^{k+1}$.

So no matter the last computation gate the induction hypothesis still holds at depth $k+1$ therefore $d_k \leq 2^{k+1}$ and $c_k \leq 2^{\frac{(k+1)^2+3(k+1)}{2}}$. This concludes the proof. ◀

4 Problems not in $\mathsf{NC}^{\mathsf{alg}}$

In this section using theorem 2 we will present different problems not in $\mathsf{NC}^{\mathsf{alg}}$, including the maxflow problem. As a warm up we will begin by showing that the the mod 2 function (or bit extraction as Mulmuley calls it in his paper) cannot be computed by algebraic circuits of small depth. But first let us properly define what we mean by computing a function.

In the beginning of this paper we have proved that $\mathsf{NC}^{\mathsf{alg}}$ is at least as expressive as iPRAM when only considering integer inputs. Therefore given a family of functions f_n and a family of intervals D_n such that $f_n : D_n \mapsto \mathbb{R}$ there are two ways to meaningfully analyze its computability in $\mathsf{NC}^{\mathsf{alg}}$. The first one is the obvious one: is there a sequence of circuits $C_n \in \mathsf{NC}^{\mathsf{alg}}$ such that $\forall \lambda \in D_n, C_n(\lambda) = f_n(\lambda)$. The second one focuses only on integers inputs: is there a sequence of circuits $C_n \in \mathsf{NC}^{\mathsf{alg}}$ such that $\forall k \in \mathbb{N} \bigcap D_n, C_n(k) = f_n(k)$. If a sequence of function cannot be computed on the integers by circuits of $\mathsf{NC}^{\mathsf{alg}}$ then it cannot be computed by iPRAM and it surely cannot be computed on the whole of D_n. When tackling problems we will often consider both version of computability.

Here is the general scheme for proving that a decision problem (the function returns either 0 or 1 depending on membership in a specified language) is not computed by an algebraic circuit of low depth. Take a subset L of $\mathbb{R}^n$ which will be the decision problem and restrict your view to $[0, 2^m]^n$, n is to be thought as the number of inputs and m as the bitlength of the integers we work with. Often in problems we only consider n and take m to be polynomial in n, it makes sense as in the bit world we may in polynomial time, only work with at most polynomial bit length integers. If L is complex enough the hope is that there will a a lot of alternations inside the hypercube $[0, 2^m]^n$ between areas belonging to L and areas not belonging to L. For instance if $n = 1$ (which is the only case studied by theorem 2), L is complex if $[0, 2^m] \bigcap L$ is equal to the union of exponentially many disjoint

intervals (exponential in m)[8]. Indeed in that case L cannot be recognized (meaning returning 1 when the output is in L, 0 otherwise) by an algebraic circuit of depth polylogarithmic in m because those cannot compute piece-wise polynomial fraction with exponentially many pieces. When n is not equal to 1 we need not worry, one must simply find an integer c and a one dimensional curve $f : [0; 2^{m^c}] \mapsto [0; 2^m]^n$ called parametrization, such that as λ ranges across $[0; 2^{m^c}]$, f goes through many alternations of L and L^C (the complementary of L), and such that f is expressible using a circuit of $\mathsf{NC}^{\mathsf{alg}}$. Then no circuit of $\mathsf{NC}^{\mathsf{alg}}$ may recognize L, otherwise a circuit for L composed with a circuit for f would yield a piece-wise polynomial function with exponentially many pieces. In our proof for maxflow we use a linear parametrization, but in general it does not need be linear. This parametrization trick may not work in some instances, meaning there probably exists problems over $\mathbb{R}^n$ not in $\mathsf{NC}^{\mathsf{alg}}$ but such that no parametrization goes through exponentially many alternation of L and L^C. For instance, in the decision version of the maxflow problem we fail to come up with a proper parametrization to show that it does not belong to $\mathsf{NC}^{\mathsf{alg}}$. Is it because such parametrization does not exist or because we could not find it ? We do not know. Our guess is that proving that the decision version of maxflow is not in $\mathsf{NC}^{\mathsf{alg}}$, is most likely provable, if not using parametrization, using tech.

4.1 Bit extraction

We start with proving that the mod 2 function cannot be computed by a circuit of small depth. Although the mod 2 function can be defined on $\mathbb{R}$ we only look at the mod 2 function on $[\![0; 2^m]\!](= [0; 2^m] \cap \mathbb{N})$ and we prove that on integer inputs no circuit of depth polylog(m) may always agree with the mod 2 function.

The next lemma states that no piece-wise polynomial fraction with few pieces or low augmented degree can agree with the mod 2 function on many consecutive integers.

▶ **Lemma 1.** *Let f be a piece-wise polynomial fraction over $\mathbb{R}$, if $\forall k \in [\![0; N]\!], f(k) = k\%2$ then $d(f) * c(f) \geq \frac{N}{10}$*

Proof. Let f be a piece-wise polynomial fraction as described in the theorem and call c its number of pieces, $I_1, \ldots, I_c$ its pieces. Assume c is less than $N/10$ (otherwise the theorem is proved), there must be a piece I_j containing at least N/c integers of $[\![0; N]\!]$. We remind that over I_j f is a polynomial fraction which we note $\frac{P}{Q}$, since $k \mapsto k\%2$ has $N/2c$ zeroes over I_j, so must P, P also cannot be the zero function because $k \mapsto k\%2$ is equal to 1 for some integers in I_j, therefore P is of degree at least $N/2c$. Therefore $d(f) * c \geq (N/2c) * c \geq N/10$. ◀

The next theorem states that no circuit of polylog(n) depth may compute the mod 2 function (we only consider the first 2^n integer inputs).

▶ **Theorem 3.** *Let c and d be two positive real numbers, for any function family f_n computed by a circuit family of depth $c \log^d(n)$, there exists $N \in \mathbb{N}$ such that $\forall n > N, \exists k \in [\![0; 2^n]\!], f_n(k) \neq k\%2$.*

Proof. Let $(C_n)_n$ be a sequence of circuits of depth $k_n = O(c \log^d(n))$. Since $n \mapsto n\%2$ is a function with one input, we may consider $(C_n)_n$ to be a sequence of circuits with one input. For any n, the function computed by C_n is a piece-wise polynomial fraction f. By theorem 2 we have that $c(f) \leq 2^{\frac{k_n^2+3k_n}{2}}$ and $d(f) \leq 2^{k_n}$ therefore $c(f)d(f) = o(2^n)$. Therefore for large enough n, using Lemma 1, f cannot coincide with $n \mapsto n\%2$ for all integers in $[\![0; 2^n]\!]$. ◀

[8] actually any interval is the union of exponentially many disjoint ones, so more precisely we should require that L is not equal to the union of less than exponentially many intervals

It was already known to Mulmuley that the problem of computing the last bit was a hard problem for $\mathsf{NC}^{\mathsf{alg}}$, from our understanding and if we recap his views informally : he dismisses it because there is no polynomial-depth circuit for this problem either[9], so it is not so interesting. Indeed he considers that the size of this problem is 1 : we are given 1 number and are asked to return its last bit. Ìn theorem 4 prove that a very similar set has no circuits in $\mathsf{NC}^{\mathsf{alg}}$ yet it has polynomial-depth circuits.

▶ **Theorem 4.** *The set*

$$\{(x_1, \ldots, x_n) \in \mathbb{N}^n; x_n \% 2 = 0 \wedge x_n < 2^n\}.$$

has polynomially bounded arithmetic circuits which agree with it over $[[0; 2^n]]$*, but none in* $\mathsf{NC}^{\mathsf{alg}}$*.*

Proof. Let us call A the set of the theorem. One can check that x_n is less than 2^n and if so retrieve with a circuit of size linear in n the last bit of x_n (using standard dichotomic search). So set A has polynomially bounded circuits.

On the other hand setting all x_i but x_n to 0 we already know that the set

$$\{(0, \ldots, 0, x_n) \in \mathbb{N}^n; x_n \% 2 = 0 \wedge x_n < 2^n\}.$$

is not in $\mathsf{NC}^{\mathsf{alg}}$ by a proof similar to the one of theorem 3. Therefore the set A is not in $\mathsf{NC}^{\mathsf{alg}}$ either. ◀

Now we would like to take a moment to ponder over what the last theorem means with regards to the algebraic approach. When presented with Mulmuley's original result we were told this three-part story: One wants to prove that $P \neq NC$. Maxflow is P-complete, so we just need to prove that Maxflow is not in NC. It is too hard, but fortunately we can prove that the algebraic version of Maxflow is not in the algebraic version of NC. Surely this has drawn us closer to proving $P \neq NC$? Well it seems to us that the last theorem dampens that assumption, we have an example of a trivial problem for bit complexity classes which is not in $\mathsf{NC}^{\mathsf{alg}}$ (yet has polynomial size circuits !). So then arises the question why bother with maxflow when trivial problems already are not in $\mathsf{NC}^{\mathsf{alg}}$? In our opinion, this remark is valid in the context of assessing the computational power of $\mathsf{NC}^{\mathsf{alg}}$ and thus the extent to which impossibility results are meaningful. However there are still good reasons to care about the maxflow problem : first of all, even though maxflow is a P-complete problem in the bit world, there is no obvious guarantee (at least to us) that maxflow is complete in the algebraic world, so a proof that a trivial problem is not in $\mathsf{NC}^{\mathsf{alg}}$ could not entail anything regarding maxflow, secondly it is interesting in its own right to know if maxflow can be solved fast in parallel using only algebraic operations.

4.2 Bijections from $\mathbb{N}$ to $\mathbb{N}^2$

Before going on to the maxflow problem we focus on the problem of computing a bijection between $\mathbb{N}$ and $\mathbb{N}^2$. If our model could do this we could reduce problems over n inputs to problems over 1 input by repeatedly applying the bijection (the same way in the bit world problems over $\mathbb{N}^2$ might as well be problems over $\mathbb{N}$ by associating a number to each pair of number). We do not directly prove that no bijection exists however we prove that we cannot inverse the usual Cantor pairing function. Using similar arguments one may be able to prove that no bijection between $\mathbb{N}$ and $\mathbb{N}^2$ may be computed by a circuit of low depth.

[9] we note : unless the bitlength is part of the size

▶ **Theorem 5.** *Let $f : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$ be the usual Cantor pairing function $f(n,m) = \frac{1}{2}(m+n)(m+n+1)+m$. This function is bijective and its reciprocal f^{-1} cannot be computed over $[\![0; 2^n]\!]$ by an algebraic circuit of depth polylogarithmic in n .*

Proof. Consider the function $g(x,y) = (x < y)$. (it returns either 0 or 1) The function $g(f^{-1})$ has $2^{\frac{n}{2}}$ alternations between 0 and 1 over $[\![0; 2^n]\!]$ therefore it cannot be computed by a circuit of $\mathsf{NC}^{\mathsf{alg}}$ (by arguments used to prove theorem 1 and 3) . But g can be computed by a circuit of $\mathsf{NC}^{\mathsf{alg}}$ ($<$ is a gate of $\mathsf{NC}^{\mathsf{alg}}$) therefore it must be f^{-1} which cannot be computed by a circuit of $\mathsf{NC}^{\mathsf{alg}}$. ◀

4.3 Maxflow

Now we focus our attention to the maxflow problem. In the maxflow problem we are given a network (an directed graph) of edges with specified capacities, a threshold k, a two specified nodes s and t, the question then asked is can a flow of size k flow from s to t.

▶ **Definition 12** (Maxflow problem)**.** *An instance of an **algebraic maxflow problem** of size n^2 is a directed graph with n nodes with each edge (as much as n^2) labelled with a real number called capacity. It is represented by its adjacency matrix representation. We say that a circuit A solves the **algebraic maxflow problem** on size n^2 if on any instance of size n^2 it returns the maximal flow from node 1 to node n. We say that a circuit A solves the **integer maxflow problem** on size n^2 if on any instance containing only integers it returns the maximal flow from node 1 to node n*

▶ **Definition 13** (Decision maxflow problem)**.** *An instance of a **decision maxflow problem** of size n^2 is an undirected graph with n nodes with each edge (as much as n^2) labelled with a real number called capacity and a threshold k. The graph is represented by its adjacency matrix representation. We say that a circuit A solves the **decision maxlow problem** on size n^2 if on any instance of size n^2 it returns 1 if the maximal flow from node 1 to node n is greater than k, 0 otherwise. We say that a circuit A solves the **integer decision maxflow problem** on size n^2 if on any instance containing only integers it returns whether the maximal flow from node 1 to node n is greater than k.*

This gives us 4 variants of the maxflow problem. We can of course further limit the complexity of these problem by only requiring algorithms solving them to work on an interval D_n instead of the whole of $\mathbb{R}$. In his paper [2] Mulmuley proves that all 4 cannot be solved by circuits of iPRAM even if we only consider polynomial size bit lengths integer, i.e $D_n = [0, 2^{n^c}]$. We will now give a simple proof that the algebraic maxflow problem cannot be solved by circuits of $\mathsf{NC}^{\mathsf{alg}}$. We introduce for that the linear parametrization version of the maxflow problem. For a given size n^2, each capacity c is now an affine function of parameter λ where the coefficients a and b are integers of bitlength polynomial in n ($c = a + b * \lambda$). Given $p, q \in \mathbb{Z}^2$ this defines for all $\lambda \in [p, q]$ an instance $G(\lambda)$ of the maxflow problem. Let I be the function such that $I(G(\lambda))$ returns the maximal flow from node 1 to node n. It has been proven in [1] and [2] that there is a family $G_n(\lambda)$ of linearly parametrized graph with polynomial-bit-length parameters and $O(n^2)$ edges, such that $I(G_n(\lambda))$ has $O(2^n)$ breakpoints (slopes changes) over $[0, T]$ with $T = 2^{n+1}$.

▶ **Theorem 6.** *The algebraic maxflow problem cannot be computed by circuits of* $\mathsf{NC}^{\mathsf{alg}}$

Proof. Let us suppose there is a family of circuits $C_n \in \mathsf{NC}^{\mathsf{alg}}$ such that C_n computes the algebraic maxflow problem for size n^2 graphs. Then C_n must return the correct value on $G_n(\lambda)$ for all λ. Consider B_n a circuit of depth 1 and $O(n^2)$ largeness taking λ as input

and outputting $G_n(\lambda)$ (represented via its adjacency matrix). Then the circuit $C_n(B_n(\lambda))$ is a circuit of $\mathsf{NC}^{\mathsf{alg}}$ with one input computing $I(G_n(\lambda))$ which is a linear function with 2^n breakpoints, but such a function cannot be computed by a circuit of $\mathsf{NC}^{\mathsf{alg}}$ by theorem 2, contradiction. ◀

▶ **Theorem 7.** *The integer algebraic maxflow problem cannot be computed by circuits of* $\mathsf{NC}^{\mathsf{alg}}$

Proof. We sketch an idea of proof only. Consider the construction of Mulmuley, notice that the breakpoints of $I(G_n(\lambda))$ are more than distance 2 away from each other, scale everything up by a factor of 2^n (one may create the integer 2^n in $log(n)$ depth circuits using repeated squaring) so that now the breakpoints are 2^n away from each other, this gives you a function $I'(\lambda)$ which is piece wise linear with each piece containing 2^n integers, argue that any piece-wise polynomial fraction which agrees with I' must have a large number of pieces or a large augmented degree (like in Lemma 1), then conclude. ◀

The last remaining piece remaining is the decision version, but we come across a slight hiccup: solvability of the decision version of a problem means being able to compute its epigraph, indeed the decision version asks us to compute $(x, y) \mapsto (y < f(x))$. Intuitively speaking it would be very strange if we could in our algebraic setting compute the epigraph of function but not its graph. The solvability of the decision version should entail the solvability of the search version (up to a given precision) by way of dichotomic search: if we can compute $(x, y) \mapsto (y < f(x))$ then we can approximatively compute $x \mapsto f(x)$ in a similar time (for instance the time is multiplied by $\log(f(x))$ if we want to be precise up to the closest integer). Unfortunately in $\mathsf{NC}^{\mathsf{alg}}$ for functions outputting values exponential in n, we cannot reduce the complexity of the decision version to that of the search version via dichotomic search (we would need a depth $\log(2^n) = n$ but we only have polylog(n) depth). Looking closer at the construction of Mulmuley in [2], the function $I(G_n(\lambda))$ does output for most $\lambda \in [0, 2^n]$ values bigger than $O(2^n)$.

We now give leads to tackle what looks to us to be only a technical issue. The most promising idea would be to find a family of function f_n computable with low depth circuits and close enough to $I(G_n(\lambda))$, i.e. such that the difference $f_n(\lambda) - I(G_n(\lambda)) = O(n^d)$ for many inputs, we could then use dichotomic search to find the value of $I(G_n(\lambda))$. This might actually be possible because looking closer at Mulmuley's construction $I(G_n(\lambda))$ vaguely grows in $\lambda^{\log(n)}$ which is a polynomial in λ (we can hardcode $\log(n)$ for every n), so an algebraic circuit of low depth may compute it. The technical part is properly analyzing the growth rate of the function $I(G_n(\lambda))$.

In case this fails another idea would be to fiddle around a bit with Mulmuley's construction to directly deal with the decision version. Yet another approach would be to analyse circuits of $\mathsf{NC}^{\mathsf{alg}}$ with 2 inputs but it would certainly complicate the mathematics. Finally there is hope that for non pathological functions f we may prove in general that uncomputability of f entails the uncomputability of $(y, x) \mapsto y < f(x)$.

References

1 Patricia June Carstensen. *The Complexity of Some Problems in Parametric Linear and Combinatorial Programming.* PhD thesis, University of Michigan, USA, 1983. AAI8314249.

2 Ketan Mulmuley. Lower bounds in a parallel model without bit operations. *SIAM Journal on Computing*, 28(4):1460–1509, 1999. doi:10.1137/S0097539794282930.

3 Thomas Seiller, Luc Pellissier, and Ulysse Léchine. Unifying lower bounds for algebraic machines, semantically. quoicoubeh, November 2022. URL: https://hal.science/hal-01921942.

Solving Odd-Fair Parity Games

Irmak Sağlam
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Anne-Kathrin Schmuck
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

This paper discusses the problem of efficiently solving parity games where player Odd has to obey an additional *strong transition fairness constraint* on its vertices – given that a player Odd vertex v is visited infinitely often, a particular subset of the outgoing edges (called *live edges*) of v has to be taken infinitely often. Such games, which we call *Odd-fair parity games*, naturally arise from abstractions of cyber-physical systems for planning and control. In this paper, we present a new Zielonka-type algorithm for solving Odd-fair parity games. This algorithm not only shares *the same worst-case time complexity* as Zielonka's algorithm for (normal) parity games but also preserves the algorithmic advantage Zielonka's algorithm possesses over other parity solvers with exponential time complexity.

We additionally introduce a formalization of Odd player winning strategies in such games, which were unexplored previous to this work. This formalization serves dual purposes: firstly, it enables us to prove our Zielonka-type algorithm; secondly, it stands as a noteworthy contribution in its own right, augmenting our understanding of additional fairness assumptions in two-player games.

2012 ACM Subject Classification Theory of computation → Solution concepts in game theory

Keywords and phrases parity games, strong transition fairness, algorithmic game theory

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.34

Related Version *Extended Version*: `https://arxiv.org/abs/2307.13396` [37]

Funding This work was partially supported by the DFG projects SCHM 3541/1-1 and 389792660 TRR 248–CPEC.

Acknowledgements We are grateful for the immense support provided by Munko Tsyrempilon for the experimental validation.

1 Introduction

Parity games are a canonical representation of ω-regular two-player games over finite graphs, which arise from many core computational problems in the context of correct-by-construction synthesis of reactive software or hardware. In particular, two player games on graphs have been extensively used in the context of cyber-physical system design [41, 6], showing their practical importance. *Fairness*, on the other hand, is a property that widely occurs in this context - both as a desired property to be enforced (e.g., requiring a synthesized scheduler to fairly serve its clients), as well as a common assumption on the behavior of other components (i.e., assuming the network to always eventually deliver a data packet). While *strong fairness* encoded by a Streett condition necessarily incurs a high additional cost in synthesis [14], it is known that the general reactivity(1) (GR(1)) fragment of linear temporal logic (LTL) [7] allows for efficient synthesis in the presence of very restricted fairness conditions. Due to its efficiency, it is extensively used in the context of cyber-physical system design, e.g. [45, 1, 30, 26, 27, 40].

Despite the omnipresence of fairness in such synthesis problems and the success of the GR(1) fragment, not much else is known about tractable fairness constraints in synthesis via two player games on graphs. A notable exception is the recent work by Banerjee et.

© Irmak Sağlam and Anne-Kathrin Schmuck;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 34; pp. 34:1–34:24
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

al. [5] which considers the sub-class of *strong* ***transition*** *fairness assumptions* [35, 15, 4] which require that whenever the environment player vertex v is visited infinitely often, a particular subset of the outgoing edges (called *live edges*) of v has to be taken infinitely often. In other words, *strong* ***transition*** *fairness assumptions* limit *strong fairness assumptions* to individual transitions. Despite their limited expressive power, such restricted fairness constrains do naturally arise in resource management [8], in abstractions of continuous-time physical processes for planning [9, 10, 34, 11, 36, 2] and controller synthesis [42, 32, 29], which makes them interesting to study.

Concretely, Banerjee et. al. [5] show that *parity games* with strong transition fairness assumptions on player Odd – which we call *Odd-fair parity games* – can be solved via a symbolic fixed-point algorithm in the μ-calculus with almost the same computational worst case complexity as the algorithm for the "normal" version of the same game. The existence of quasi-polynomial time solution algorithms for Odd-fair parity games then follows as a corollary of their nested fixed-point characterization [17, 3, 20]. Unfortunately, it is well known that symbolic fixed-point computations become cumbersome very fast for parity games, as the number of priorities in the game graph increases, leading to high computation times in practice. Given the known inefficiency of existing quasi-polynomial algorithms for parity games [44, 33], despite their theoretical advantages, they are not viable candidates for adoption in the development of efficient solution algorithms for Odd-fair parity games either. For (normal) parity games, computational tractability can be achieved by other algorithms, such as Zielonka's algorithm [46], tangle learning [43] or strategy-improvement [38], implemented in the state-of-the-art tool `oink` [44], with Zielonka's algorithm being widely recognized as the most prominent approach.

The **main contribution** of this paper is a Zielonka-type algorithm, referred to as "*Odd-fair Zielonka's algorithm*", for solving Odd-fair parity games. This novel algorithm meets the efficiency of Zielonka's algorithm while maintaining the same computational worst-case complexity (which is exponential just like the worst-case complexity of the fixed-point algorithm from [5]). Using a prototype implementation, we experimentally verify its efficiency, demonstrating that it matches Zielonka's algorithm in speed, thereby highlighting its comparable performance to fixed-point algorithms for classical parity games.

In contrast to the work by Banerjee et. al. [5], the adaptation and the correctness proof of Odd-fair *Zielonka's algorithm* requires the understanding of Odd player strategies, while [5] studies the solution of such games solely from the Even player's perspective. Unfortunately, Odd strategies are substantially more complex than Even strategies in such games, as they are not positional – while player Even strategies still are (see [5, Thm.3.10]). The **second contribution** of this paper is therefore the formalization of Odd player strategies in Odd-fair parity games, via so called *strategy templates*, which was unexplored prior to this work. We give a constructive proof for the existence of strategy templates winning for Odd from all vertices in the winning region of Odd. This serves dual purposes: firstly, it enables us to prove the correctness of the Odd-fair Zielonka's algorithm; secondly, it stands as a noteworthy contribution in its own right, augmenting our understanding of additional fairness assumptions in two-player games which are currently only unsatisfactorily adressed in various practically motivated synthesis problems.

2 Preliminaries

Notation. We use $\mathbb{N}$ to denote the set of natural numbers including zero and $\mathbb{N}^+$ to denote positive integers. Let Σ be a finite set. Then Σ^* and Σ^ω denote the sets of finite and infinite words over Σ, respectively.

Game graphs. A *game graph* is a tuple $G = \big(V, V^0, V^1, E\big)$ where (V, E) is a finite directed graph with *edges* E and *vertices* V partioned into player 0 and player 1 vertices, V^0 and V^1, respectively. Without loss of generality, we can assume that all nodes in V have at least one outgoing edge. Under this assumption, there exist plays from each vertex. A *play* originating at a vertex v_0 is an infinite sequence of vertices $\pi = v_0 v_1 \ldots \in V^\omega$. For $v \in V$, $E(v)$ denotes its successor set $\{w \mid (v, w) \in E\}$.

LTL winning conditions. Given a game graph G, we consider winning conditions specified using a formula Φ in *linear temporal logic* (LTL) over the vertex set V, that is, we consider LTL formulas whose atomic propositions are sets of vertices. In this case the set of desired infinite plays is given by the semantics of Φ which is an ω-regular language $\mathcal{L}(\Phi) \subseteq V^\omega$. The standard definitions of ω-regular languages and LTL are omitted for brevity and can be found in standard textbooks [4]. A game graph G under the winning condition Φ is written as $\langle G, \Phi\rangle$. A play π is winning for player 0 in $\langle G, \Phi\rangle$ if $\pi \in \mathcal{L}(\Phi)$, i.e. $\pi \models \Phi$.

Strategies. A *strategy* for player j over the game graph G is a function $\rho^j : V^* \cdot V^j \to V$ with the constraint that for all $u \cdot v \in V^* \cdot V^j$ it holds that $\rho^j(u \cdot v) \in E(v)$. A play $\pi = v_0 v_1 \ldots \in V^\omega$ is compliant with ρ^j if for all $i \in \mathbb{N}$ holds that $v_i \in V^j$ implies $v_{i+1} = \rho^j(v_0 \ldots v_i)$. A strategy ρ^j is winning from a subset V' of vertices of the game $\langle G, \Psi\rangle$ if all plays π in G that start at a vertex in V' and are compliant with ρ^j are winning w.r.t. Ψ. A strategy ρ is called *positional* iff for all $w_1, w_2 \in V^*$, $\rho(w_1 \cdot v) = \rho(w_2 \cdot v)$.

Parity Games. Parity games are particular two player games over a game graph G where the winning condition is given by a particular mapping of vertices. Formally, a parity game is a tuple $\mathcal{G} = \langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle$, where $(V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E)$ is a game graph and $\chi : V \to \mathbb{N}^+$ is a function which labels each vertex with an integer value, called a *priority*. The players 0 and 1 are called Even and Odd in a parity game and a play $\pi = v_1 v_2 \ldots$ is winning for Even iff $\max\{\inf(\pi)\}$ is *even*, where $\inf(\pi)$ is the set of vertices visited infinitely often in π. Otherwise the play is winning for Odd.

A node $v \in V$ is said to be won by Even, if Even has a (winning) strategy ρ such that all plays $\pi = v \cdot \pi'$ that are compliant with ρ are won by Even. The winning region of Even is the set of all nodes won by Even and is denoted by $\mathcal{W}_{\mathsf{Even}}$. The winning region of Odd, $\mathcal{W}_{\mathsf{Odd}}$, is defined similarly. It is well-known that parity games are determined, that is, all nodes are either in $\mathcal{W}_{Even}$ or in $\mathcal{W}_{Odd}$; and that both players have positional winning strategies from their respective winning regions [12].

Odd-Fair Parity Games. An *Odd-fair parity game* $\mathcal{G}^\ell$ is a tuple $\langle \mathcal{G}, E^\ell\rangle$, where $\mathcal{G} = \langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle$ is a parity game, $E^\ell \subseteq E$ is a set of *live edges* that originate from Odd player vertices and $V^\ell \subseteq V_{\mathsf{Odd}}$, the domain of the relation E^ℓ, is the set of *live vertices*. The live edges induce a *strong transition fairness constraint* – whenever a live vertex v is visited infinitely often, every outgoing live edge $(v, w') \in E^\ell$ needs to be taken infinitely often. Formally, a play π in $\mathcal{G}$ *complies* with E^ℓ if the LTL formula[1]

$$\alpha := \bigwedge_{(v,w)\in E^\ell} (\Box\Diamond v \implies \Box\Diamond(v \wedge \bigcirc w)) \tag{1}$$

[1] Here, $\Box$, $\Diamond$ and $\bigcirc$ stand for the LTL operators 'always', 'eventually' and 'next'.

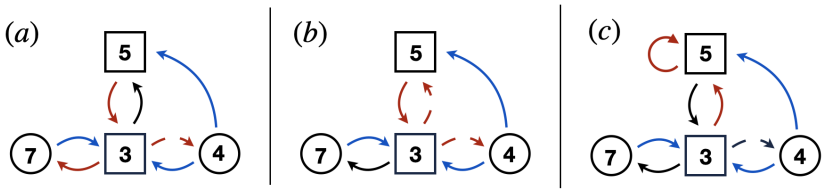


Figure 1 Odd-fair games with player even V_{Even} (circles) and player odd V_{Odd} (squares) vertices (labeled with their priorities). Live edges E^{ℓ} (dashed) originate from V_{Odd}. Colored player Odd (red) and player Even (blue) edges belong to player Odd's strategy template.

holds along π, i.e. $\pi \models \alpha$. A play π is winning for Even in $\mathcal{G}^{\ell}$ if and only if $\pi \models \neg\alpha$ or $\max\{\inf(\pi)\}$ is *even*. Dually, π is winning for Odd iff $\pi \models \alpha$ and $\max\{\inf(\pi)\}$ is odd. A strategy ρ over G is therefore winning for Even (resp. Odd) in $\mathcal{G}^{\ell}$ if all plays compliant with ρ are winning for Even (resp. Odd) in $\mathcal{G}^{\ell}$.

As the winning condition of a parity game can be equivalently modeled by a suitably defined LTL winning condition, we see that Odd-fair parity games are a special ω-regular game with perfect information. This implies that Odd-fair parity games are determined (by the Borel determinacy theorem [31]) and whenever there exists a winning strategy for Even/Odd in such a game, then there also exists one with *finite* memory [16].

3 Strategy Templates

In this section, we introduce a formalization of player Odd strategies in Odd-fair parity games via *strategy templates*. In contrast to player Even, player Odd winning strategies are no longer positional in Odd-fair parity games, as illustrated by the following example.

▶ **Example 1.** Consider the three different parity games depicted in Fig. 1. In all three games, Odd has a winning strategy from all vertices, i.e., $\mathcal{W}_{Odd} = V$. However, in order to win, the vertex 3 has to be seen infinitely often in game (a) and (b), which forces Odd to use its live edge\s infinitely often. This prevents the existence of a positional strategy for Odd in games (a) and (b): In (a) it needs to somehow alternate between (it's only) live edge to 4 and a "normal" edge to 7 (both indicated in red) in order to win, and in (b) it needs to somehow alternate between all its live edges (also indicated in red). In the game (c), Odd can win by 'escaping' its live vertex 3 to a "normal" vertex 5, and thereby has a positional strategy.

Now consider the subgraph of each game formed by all colored edges (red and blue), which include the strategy choices from V_{Odd} and *all* outgoing edges from V_{Even}. As we have seen that Odd needs to play all red edges repeatably, this subgraph represents the paths that *can* be seen in the game depending on the Even strategy. Hence, a node $v \in V^{\ell} \subseteq V_{\mathsf{Odd}}$ can be seen infinitely often in a play (compliant with Odd's strategy), if it lies on a cycle in this subgraph. We observe that, in games (a) and (b), node 3 lies on cycles in this subgraph, whereas in game (c), it does not. We further see that whenever a vertex $v \in V^{\ell}$ lies on a cycle, Odd needs to take all its outgoing live edges (as for vertex 3 in example (b)) and possibly one more edge (as for vertex 3 in example (a)), for all other vertices in V_{Odd} a positional strategy suffices (as for vertex 5 in all examples, and for vertex 3 in example (c)). This shows that Odd strategies are intuitively still "almost positional".

The intuitions conveyed by Ex. 1 are formalized by the following definitions.

▶ **Definition 2** (Odd Strategy Template). *Given an Odd-fair parity game $\mathcal{G}^\ell = \langle \mathcal{G}, E^\ell \rangle$ with $\mathcal{G} = \langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi \rangle$, an Odd* strategy template *$\mathcal{S}$ over $\mathcal{G}^\ell$ is a subgraph of $\mathcal{G}$ given as follows: $\mathcal{S} := (V', E')$ where $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$ such that the following hold,*

- *if $v \in V_{\mathsf{Odd}} \cap V'$ does not lie on a cycle in (V', E'), then $|E'(v)| = 1$,*
- *if $v \in V_{\mathsf{Odd}} \cap V'$ lies on a cycle in (V', E') then $E^\ell(v) \subseteq E'(v)$ and $1 \leq |E'(v)| \leq |E^\ell(v)|+1$,*
- *if $v \in V_{\mathsf{Even}} \cap V'$, then $E'(v) = E(v)$.*

▶ **Definition 3.** *Let $\mathcal{G}^\ell = \langle \mathcal{G}, E^\ell \rangle$ be an Odd-fair parity game with Odd strategy template $\mathcal{S} = (V', E')$, and $V'_{\mathsf{Odd}} := V' \cap V_{\mathsf{Odd}}$. Then an Odd strategy ρ is said to be* ***compliant*** *with $\mathcal{S}$ if it is a winning strategy in the game $\langle G, \alpha' \rangle$ where $G = (V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E)$ and*

$$\alpha' := \bigwedge_{v \in V'_{\mathsf{Odd}}} (\Box (v \implies \bigvee_{(v,w) \in E'} \bigcirc w)) \tag{2a}$$

$$\wedge \bigwedge_{v \in V'_{\mathsf{Odd}}} (\Box \Diamond v \implies \bigwedge_{(v,w) \in E'} \Box \Diamond (v \wedge \bigcirc w)). \tag{2b}$$

Intuitively, for all Odd vertices in $\mathcal{S}$, the strategy ρ compliant with $\mathcal{S}$ takes only their outgoing edges in $\mathcal{S}$ (2a), and if a play visits an Odd node v infinitely often, then ρ takes each of v's outgoing edges in $\mathcal{S}$ infinitely often (2b). For an Odd strategy template $\mathcal{S}$, if $v \in V'_{\mathsf{Odd}}$ lies on a cycle in $\mathcal{S}$, then by Def. 2, $\mathcal{S}$ contains all live outgoing edges of v. By (2b) any Odd strategy ρ compliant with $\mathcal{S}$ satisfies the fairness condition in (1) for v. On the other hand, if $v \in V'_{\mathsf{Odd}}$ does not lie on a cycle in $\mathcal{S}$, then by (2a) any such ρ sees v at most once. Thus ρ trivially satisfies (1) for v. This observation is stated in the following proposition.

▶ **Proposition 4.** *Given the premisses of Def. 3 let π be a play starting from a node in V' that complies with ρ. Then $\pi \models \alpha$ where α is the LTL formula in* (1).

Next, we define Even strategy templates. Each Even strategy template encodes a unique Even positional strategy, which is known to exist in Odd-fair parity games [23], due to the lack of fair edges defined on Even vertices.

▶ **Definition 5.** *Given an Odd-fair parity game $\mathcal{G}^\ell = \langle \mathcal{G}, E^\ell \rangle$ with $\mathcal{G} = \langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi \rangle$, an Even* strategy template *$\mathcal{S}$ over $\mathcal{G}^\ell$ is a subgraph of $\mathcal{G}$ given as $\mathcal{S} := (V', E')$ where $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$ such that,*

- *if $v \in V_{\mathsf{Even}} \cap V'$, then $|E'(v)| = 1$,*
- *if $v \in V_{\mathsf{Odd}} \cap V'$, then $E'(v) = E(v)$.*

An Even strategy ρ is compliant with the Even strategy template $\mathcal{S} = (V', E')$ if for all $v \in V'_{\mathsf{Even}}$, $\rho(v) = E'(v)$. In other words, ρ is the positional strategy defined by $\mathcal{S}$.

Let ρ be an Odd (Even) strategy, compliant with the Odd (Even) strategy template $\mathcal{S}$ and let π be a play compliant with ρ. Then we call π a play *compliant with $\mathcal{S}$*.

▶ **Definition 6.** *An Odd (Even) strategy template $\mathcal{S} = \langle V', E' \rangle$ is* winning *in the Odd-fair parity game $\mathcal{G}^\ell$ if all Odd (Even) strategies ρ compliant with $\mathcal{S}$ are winning for player Odd (Even) in $\mathcal{G}^\ell$ from V'. A winning Odd (Even) strategy template $\mathcal{S}$ is called* maximal *if $V' = \mathcal{W}_{\mathsf{Odd}}$ ($\mathcal{W}_{\mathsf{Even}}$).*

We note that maximal winning Odd (Even) strategy templates $\mathcal{S}$ immediately imply that for every vertex $v \in \mathcal{W}_{\mathsf{Odd}}$ ($\mathcal{W}_{\mathsf{Even}}$) there exists a winning strategy for player Odd (Even) from v that is compliant with $\mathcal{S}$. The existence of maximal winning Even strategy templates follows from the existence of positional Even strategies [23]. The first main contribution of this paper is a constructive proof showing the existence of maximal winning Odd strategy templates given in the next section. This result is then used in Sec. 5 to prove the correctness of Odd-fair Zielonka's algorithm, which is introduced there.

4 Existence of Maximal Winning Odd Strategy Templates

This section proves the existence of maximal winning Odd strategy templates[2] in Odd-fair parity games, formalized in the following theorem.

▶ **Theorem 7.** *Given an Odd-fair parity game $\mathcal{G}^\ell$, there exists a maximal winning Odd strategy template.*

We prove Thm. 7 by giving an algorithm which constructs $\mathcal{S}$ from a ranking function induced by a fixed-point algorithm in the μ-calculus which computes $\mathcal{W}_{\mathsf{Odd}}$. Towards this goal, Sec. 4.1 first introduces necessary preliminaries, Sec. 4.2 gives the fixed-point algorithm to compute $\mathcal{W}_{\mathsf{Odd}}$ and Sec. 4.3 formalizes how to extract a strategy template $\mathcal{S}$ from the ranking induced by this fixed-point and proves that $\mathcal{S}$ is indeed maximal and winning.

While this section uses fixed-point algorithms extensively to *construct* a maximal winning Odd strategy template towards a *proof* of Thm. 7, we note again that the proof of the new Zielonka's algorithm given in Sec. 5 only uses the *existence* of templates (i.e., the fact that Thm. 7 holds) and does not utilize their *construction* via the algorithm presented here.

4.1 Preliminaries on Fixed-Point Algorithms

This subsection contains the basic notation used in this section.

Set Transformers. Let $G = (V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E)$ be a game graph, $S, T \subseteq V$ and Λ be the player index.[3] Then we define the following predecessor operators:

$$\begin{aligned} \mathsf{Pre}^{\exists}_{\Lambda}(S) &:= \{v \in V_\Lambda \mid E(v) \cap S \neq \emptyset\} & \mathsf{Lpre}^{\exists}(S) &:= \{v \in V_{\mathsf{Odd}} \mid E^\ell(v) \cap S \neq \emptyset\} \\ \mathsf{Pre}^{\forall}_{\Lambda}(S) &:= \{v \in V_\Lambda \mid E(v) \subseteq S\} & \mathsf{Lpre}^{\forall}(S) &:= \{v \in V_{\mathsf{Odd}} \mid E^\ell(v) \subseteq S\} \end{aligned} \tag{3}$$

The predecessor operators $\mathsf{Pre}^{\exists}_{\Lambda}(S)$ and $\mathsf{Pre}^{\forall}_{\Lambda}(S)$ compute the sets of vertices with *at least one* successor and with *all* successors in S, respectively. The live predecessor operators $\mathsf{Lpre}^{\exists}(S)$ and $\mathsf{Lpre}^{\forall}(S)$ restrict this analysis to live edges. We see that

$$\neg\mathsf{Pre}^{\exists}_{\Lambda}(\neg S) = V_{\neg\Lambda} \cup \mathsf{Pre}^{\forall}_{\neg\Lambda}(S) \qquad \text{and} \qquad \neg\mathsf{Lpre}^{\exists}(\neg S) = V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(S) \tag{4}$$

where for a set $X \subseteq V$, $\neg X$ stands for $V \setminus X$. We combine the pre-operators from (3) into the combined set:[4]

$$\mathsf{Cpre}_{\Lambda}(S) := \mathsf{Pre}^{\exists}_{\Lambda}(S) \cup \mathsf{Pre}^{\forall}_{\neg\Lambda}(S) \tag{5a}$$

$$\mathsf{Apre}(S,T) := \mathsf{Cpre}_{\mathsf{Even}}(T) \cup (\mathsf{Lpre}^{\exists}(T) \cap \mathsf{Pre}^{\forall}_{\mathsf{Odd}}(S)) \tag{5b}$$

$$\mathsf{Npre}(S,T) := \mathsf{Cpre}_{\mathsf{Odd}}(T) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(T) \cup \mathsf{Pre}^{\exists}_{\mathsf{Odd}}(S)) \tag{5c}$$

The *controllable predecessor operator* $\mathsf{Cpre}_{\Lambda}(S)$ computes the set of vertices from which player Λ can force visiting S in *one* step. It immediately follows that

$$\neg\mathsf{Cpre}_{\mathsf{Even}}(\neg S) := \mathsf{Cpre}_{\mathsf{Odd}}(S). \tag{6}$$

---

[2] In the rest of this section, we will sometimes call Odd strategy templates simply, *strategy templates*, since these are the only strategy templates we will be dealing with.

[3] $\Lambda \in \{\mathsf{Even}, \mathsf{Odd}\}$ where $\Lambda = \mathsf{Even}$ implies $\neg\Lambda = \mathsf{Odd}$, and vice versa.

[4] Note that $\mathsf{Apre}(S,T)$ and $\mathsf{Npre}(S,T)$ are meaningful only when $T \subseteq S$ and $S \subseteq T$, respectively. Otherwise they are equivalent to $\mathsf{Cpre}_{\mathsf{Even}}(T)$ and $\mathsf{Cpre}_{\mathsf{Odd}}(T)$. We note that these preconditions will always be satisfied in our calculations due to the monotonicity of fixed-point computations.

The *almost-sure controllable predecessor* operator $\mathsf{Apre}(S,T)$ computes the set of states that can be controlled by Player Even to stay in T (via $\mathsf{Cpre}_{\mathsf{Even}}(T)$) as well as all Player Odd states in V^ℓ that (a) will eventually make progress towards T if Player Odd obeys its fairness-assumptions (via $\mathsf{Lpre}^\exists$) and (b) will never leave S in the "meantime" (via $\mathsf{Pre}^\forall_{\mathsf{Odd}}(S)$)). Using (4) and (6) we have $\mathsf{Npre}(S,T) := \neg\mathsf{Apre}(\neg S, \neg T)$.

Fixed-point Algorithms in the μ-calculus. The μ-calculus offers a succinct representation of symbolic algorithms (i.e., algorithms manipulating sets of vertices instead of individual vertices) over a game graph G. We omit the (standard) syntax and semantics of μ-calculus formulas (see [25]) and only discuss their evaluation on an example fixed-point algorithm given by a 2-nested μ-calculus formula of the form $Z = \mu Y.\ \nu X.\ \phi(X,Y)$, where $X, Y \subseteq V$ are subsets of vertices and μ and ν denote, respectively, the least and the greatest fixed-point. ϕ is a formula composed from the *monotone set transformers* in (3) and (5).

Given this formula, first, both formal variables X and Y are initialized. As Y (resp. X) is preceded by μ (resp. ν) it is initialized with $Y^0 := \emptyset$ (resp. $X^0 := V$). Now we first keep Y at its initial value and iteratively compute $X^k = \phi(X^{k-1}, Y^0)$ until $X^{k+1} = X^k$. At this point X saturates, denoted by X^∞. We then "copy" X^∞, to Y, i.e., have $Y^1 := X^\infty$, reinitialize $X^0 := \emptyset$, and re-evaluate $X^k = \phi(X^{k-1}, Y^1)$ with the new value of Y. This calculation terminates if Y saturates, i.e., $Y^\infty = Y^{l+1} = X^l$ for some $l \geq 0$, and outputs $Z = Y^\infty$. In order to remember all intermediate values of X we use $X^{l,k}$ to denote the set computed in the k-th iteration over X during the computation of Y^l. I.e., $Y^l = X^{l,\infty}$.

Additional Notation. We will use the letters l, m and n exclusively to denote *even* positive integers. For $a \leq b \in \mathbb{N}$, we will use the regular set symbol $[a,b]$ to denote the set of all integers between a and b, i.e., $[a,b] := \{a, a+1, \ldots, b\}$; and $[\![a,b]\!]$ to denote all the *even* integers between a and b. E.g. $[\![2,7]\!] = \{2,4,6\}$. In addition, given an Odd-fair parity game $\mathcal{G}^\ell$, we define the sets $C_i := \{v \in V \mid \chi(v) = i\}$ and $\overline{C_i} := V \setminus C_i$ to ease notation. We say $\mathcal{G}^\ell$ has the least even upper bound l if $C_l \cup C_{l-1} \neq \emptyset$ and $C_i = \emptyset$ for all $i > l$.

4.2 A Fixed-Point Algorithm for $\mathcal{W}_{\mathsf{Odd}}$

Given an Odd-fair parity game $\mathcal{G}^\ell = \langle\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell\rangle$ this section presents a fixed-point algorithm in the μ-calculus which computes the winning region $\mathcal{W}_{\mathsf{Odd}}$ of player Odd in Odd-fair parity games. It is obtained by negating the fixed-point formula computing $\mathcal{W}_{\mathsf{Even}}$ in [5], formalized in the following proposition and proven in the extended version of this work [37].

▶ **Proposition 8.** *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ it holds that $Z = \mathcal{W}_{\mathsf{Odd}}$, where*

$$Z := \mu Y_l.\ \nu X_{l-1}.\ \ldots \mu Y_2.\ \nu X_1.\ \bigcap_{j\in[\![2,l]\!]} \mathcal{B}_j[Y_j, X_{j-1}], \tag{7}$$
$$\text{where}\quad \mathcal{B}_j[\mathbf{Y},\mathbf{X}] := \left(\bigcup_{i\in[j+1,l]} C_i\right) \cup \left(\overline{C_j} \cap \mathsf{Npre}(\mathbf{Y},\mathbf{X})\right) \cup \left(C_j \cap \mathsf{Cpre}_{\mathsf{Odd}}(\mathbf{Y})\right).$$

Before utilizing (7) we illustrate its computations via an example.

▶ **Example 9.** Consider the Odd-fair parity game $\mathcal{G}^\ell$ depicted in Fig. 2 (left). Here, the name of the vertices coincide with their priorities, e.g., $C_2 = \{2a, 2b, 2c\}$. V_{Even} and V_{Odd} are indicated by circles and squares, respectively. Edges in E^ℓ are shown by dashed lines. As the least even upper bound in this example is $l = 4$,

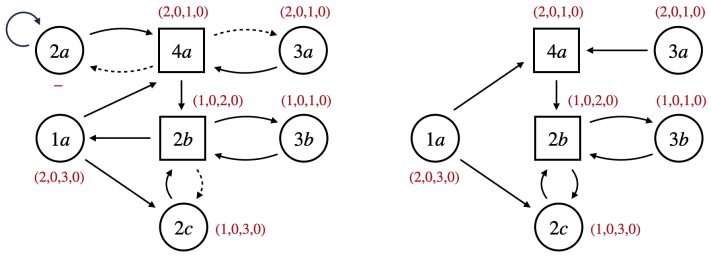


Figure 2 Odd-fair parity game $\mathcal{G}^\ell$ discussed in Ex. 9, 10, and 13 (left) and its corresponding minimum rank based maximal Odd strategy template $\mathcal{S}^{\mathcal{G}^\ell}$ as defined in Def. 12 (right).

$$
\begin{aligned}
&Z = \mu Y_4.\ \nu X_3.\ \mu Y_2.\ \nu X_1.\ \Phi^{Y_4,X_3,Y_2,X_1} \quad \text{where} \\
&\Phi^{Y_4,X_3,Y_2,X_1} := (\overline{C_4} \cap \mathsf{Npre}(Y_4, X_3)) \cup (C_4 \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_4))) \\
&\qquad \cap (\overline{C_2} \cap \mathsf{Npre}(Y_2, X_1)) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_2)) \cup C_4 \cup C_3).
\end{aligned} \tag{8}
$$

Using the notation defined in Sec. 4.1, we initialize (8) by $Y_4^0 = \emptyset$, $X_3^{0,0} = V$, $Y_2^{0,0,0} = \emptyset$ and $X_1^{0,0,0,0} = V$ and observe from (5) that $\mathsf{Cpre}_{\mathsf{Odd}}(\emptyset) = \emptyset$ and $\mathsf{Npre}(\emptyset, V) = V$. We obtain

$$
\begin{aligned}
X_1^{0,0,0,1} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,0},X_1^{0,0,0,0}} = ((\overline{C_4} \cap \mathsf{Npre}(\emptyset, V)) \cup (C_4 \cap \mathsf{Cpre}_{\mathsf{Odd}}(\emptyset))) \cap \\
&\quad ((\overline{C_2} \cap \mathsf{Npre}(\emptyset, V)) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(\emptyset)) \cup C_4 \cup C_3) \\
&= (\overline{C_4}) \cap (\overline{C_2} \cup C_4 \cup C_3) = C_3 \cup C_1 \\
X_1^{0,0,0,2} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,0},X_1^{0,0,0,1}} \\
&= C_3 \cup (C_1 \cap \mathsf{Npre}(Y_2^{0,0,0}, X_1^{0,0,0,1})) = C_3 \cup (C_1 \cap \mathsf{Npre}(\emptyset, C_3 \cup C_1)) = C_3
\end{aligned}
$$

where $\mathsf{Npre}(\emptyset, C_3 \cup C_1) = \emptyset$ as $v \in \mathsf{Npre}(\emptyset, C_3 \cup C_1)$ implies $v \in \mathsf{Cpre}_{\mathsf{Odd}}(C_3 \cup C_1) = \{2b, 4a\}$ and $v \in V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(C_3 \cup C_1)$. However, $2b, 4a$ are Odd vertices with live outgoing edges to $2a, 2c \in (V \setminus (C_3 \cup C_1))$. In the next iteration, we again get $X_1^{0,0,0,3} = C_3$ and thus X_1 saturates with C_3. Therefore, $Y_2^{0,0,1} = C_3$. Now the next round of computations of Φ results in

$$
\begin{aligned}
X_1^{0,0,1,1} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,1},X_1^{0,0,1,0}} = C_3 \cup (C_1 \cap \mathsf{Npre}(Y_2^{0,0,1}, X_1^{0,0,1,0})) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_2^{0,0,1})) \\
&= C_3 \cup (C_1 \cap \mathsf{Npre}(C_3, V)) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(C_3)) = C_3 \cup C_1 \cup \{2b\} \\
X_1^{0,0,1,2} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,1},X_1^{0,0,1,1}} = C_3 \cup \{2b\} = X_1^{0,0,1,3}
\end{aligned}
$$

Here C_1 and $\{2b\}$ get added in $X_1^{0,0,1,1}$ as $1a \in \mathsf{Npre}(C_3, V)$ trivially and $2b \in \mathsf{Cpre}_{\mathsf{Odd}}(C_3)$ due to the edge $(2b, 3b)$. C_1 is removed from $X_1^{0,0,1,2}$ since $1a$ cannot be forced by Odd to $C_1 \cup C_3 \cup \{2b\}$ in the next step. The fixed-point calculation proceeds in a similar fashion, until Y_4 reaches its saturation value $V \setminus \{2a\}$. The full computation of Z is given in [37].

4.3 Construction of a Rank-based Strategy Template

Given an Odd-fair parity game $\mathcal{G}^\ell$ with the least even priority upper bound $l \geq 0$, we define a ranking function $\mathsf{rank} : \mathcal{W}_{\mathsf{Odd}} \to \mathbb{N}^l$ first introduced in [39] and highly related to "progress measures" [24, 23, 22, 19]. Intuitively, $\mathsf{rank}(v)$ indicates in which iteration v was added to Z in (7) and never got removed from Z again, as illustrated by the following example.

▶ **Example 10.** Consider again the Odd-fair parity game depicted in Fig. 2. Here, $\mathsf{rank}(v)$ of each $v \in \mathcal{W}_{\mathsf{Odd}} = V \setminus \{2a\}$ is shown in red next to the node in the figure. Intuitively, the 4−tuple is associated with the subscript Y_4, Y_3, Y_2, Y_1 of Φ in (8). For instance $\mathsf{rank}(3a) = (2, 0, 1, 0)$ indicates that $3a$ was added to Z during the first iteration of Y_2 inside the second iteration of Y_4. More concretely, $3a \notin Y_4^0, 3a \notin Y_4^1, 3a \in Y_4^2$. So 2 is the first iteration of the Y_4 variable in which $3a$ got included in the variable. For Y_2, $3a \notin Y_2^{2,0,0}$ and $3a \in Y_2^{2,0,1}$, and therefore $\mathsf{rank}(3a) = (2, 0, 1, 0)$.

The intuition of Ex. 10 is formalized in the following definition.

▶ **Definition 11** (rank). *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ and winning region $\mathcal{W}_{\mathsf{Odd}} \subseteq V$, we define the ranking function $\mathsf{rank} : \mathcal{W}_{\mathsf{Odd}} \to \mathbb{N}^l$ for $v \in \mathcal{W}_{\mathsf{Odd}}$ such that*

$$\mathsf{rank}(v) = (r_l, 0, r_{l-1}, 0 \ldots r_2, 0) \quad \textit{if} \quad v \in \bigcap_{j \in [\![2,l]\!]} Y_j^{r_l,0,\ldots,r_j} \setminus Y_j^{r_l,0,\ldots,r_j-1}. \tag{9}$$

where the valuations of the variables Y_j are obtained from the iterations of the fixed-point calculation in (7) as illustrated in Ex. 9.

A ranking function obtained from a fixed-point computation as in (9) naturally gives rise to a positional winning strategy for the respective player in (normal) ω-regular games that allow for positional strategies. The corresponding positional strategy is obtained by always choosing a *minimum ranked successor* in the winning region.[5] We use this insight to obtain a *candidate* maximal strategy template for player Odd (which we prove to be also *winning* in Prop. 14) as follows. We start with a subgraph on $\mathcal{W}_{\mathsf{Odd}}$ defining the minimum ranked successor strategy for Odd induced by the ranking in (9), and then iteratively add all live edges of nodes that lie on a cycle in the subgraph, to the subgraph. The saturated subgraph then defines a strategy template for Odd, as formalized next.

▶ **Definition 12** (Rank-based Strategy Template). *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ on the priorities of nodes, winning region $\mathcal{W}_{\mathsf{Odd}} \subseteq V$ and the ranking function $\mathsf{rank} : \mathcal{W}_{\mathsf{Odd}} \to \mathbb{N}^l$ from Defn. 11, we define a strategy template $\mathcal{S}^{\mathcal{G}^\ell} = (\mathcal{W}_{\mathsf{Odd}}, E')$ where E' is constructed as follows:*

(S1) *for all $v \in V_{\mathsf{Even}} \cap \mathcal{W}_{\mathsf{Odd}}$, add all $(v, w) \in E$ to E';*
(S2) *for all $v \in V_{\mathsf{Odd}} \cap \mathcal{W}_{\mathsf{Odd}}$, add $(v, w) \in E$ to E' for a w with $w = argmin_{w' \in E(v)} \mathsf{rank}(w')$ (w is arbitrarily picked amongst the successors with the mimimum ranking);*
(S3) *for all $v \in V^\ell \cap \mathcal{W}_{\mathsf{Odd}}$, add all $(v, w) \in E^\ell$ to E' if v lays on a cycle in $\mathcal{S}^{\mathcal{G}^\ell}$;*
(S4) *repeat item (S3) until no new edges are added.*

We call $\mathcal{S}^{\mathcal{G}^\ell}$ the minimum rank based maximal Odd strategy template *of $\mathcal{G}^\ell$.*

▶ **Example 13.** $\mathcal{S}^{\mathcal{G}^\ell}$ for $\mathcal{G}^\ell$ from Ex. 9 is depicted in Fig. 2 (right).

It is clear from the definition that $\mathcal{S}^{\mathcal{G}^\ell}$ is an Odd strategy template in $\mathcal{G}^\ell$. It is also maximal since each $v \in \mathcal{W}_{\mathsf{Odd}}$ is assigned a rank. It remains to show that it is winning:

▶ **Proposition 14.** *Every player Odd strategy compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ is winning for Odd in $\mathcal{G}^\ell$.*

The full proof of Prop. 14 can be found in App. A.1 and we only give a proof-sketch here.

First, recall that $\mathcal{S}^{\mathcal{G}^\ell}$ is obtained by extending a minimum-rank based strategy as formalized in Def. 12. Based on this we call a play $v_1 v_2 \ldots$ in $\mathcal{S}^{\mathcal{G}^\ell}$ *minimal* if for all $v_i \in V_{\mathsf{Odd}}$, v_{i+1} is the minimum ranked successor of v_i. We further call a cycle minimal, if it is a section of

[5] See [5] for a similar construction of the positional winning strategy of Even in Odd-fair parity games

a minimal play. Now consider a play $\pi = v_0 v_1 \dots$ which is compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ and $v_0 \in \mathcal{W}_{\mathsf{Odd}}$. Since π is compliant with an Odd strategy template, it obeys the fairness condition. It is left to show that π is Odd winning. We do this by a chain of three observations,

1. If $\mathcal{W}_{\mathsf{Odd}} \neq \emptyset$, there exists a non empty set $M := \{v \in \mathcal{W}_{\mathsf{Odd}} \mid \mathsf{rank}(v) = (1, 0, 1, 0, \dots, 1, 0)\}$ (see Prop. 21).
2. All cycles in $\mathcal{S}^{\mathcal{G}^\ell}$ that pass through a vertex in M are Odd winning (see Prop. 22).
3. All infinite minimal plays in $\mathcal{S}^{\mathcal{G}^\ell}$ visit M infinitely often (see Prop. 25).

While item 1 simply follows from the observation that $(1, 0, 1, 0, \dots, 1, 0)$ is the minimum rank the ranking function assigns to a vertex and the set of nodes with this rank cannot be empty due to the monotonicity of (7), the proofs for item 2 and 3 are rather technical.

With the observations in item 1-3 being proven, we are ready to show that π is Odd winning. Observe that $\pi = v_1 v_2 \dots$ "embeds" an infinite minimal play, that is, there exists a subsequence $\pi' = v_{j_1} v_{j_2} \dots$ of π where $j_1 < j_2 < \dots$ that is a minimal play. This is because whenever a $v \in V_{\mathsf{Odd}} \cap \mathcal{W}_{\mathsf{Odd}}$ is seen infinitely often in π, $(v, v_{\min})$ is seen infinitely often as well, where $v_{\min}$ is the minimum-rank successor of v in $\mathcal{S}^{\mathcal{G}^\ell}$. Since π' visits M infinitely often (from item 3), π does so too. Then due to pigeonhole principle, there exists an $x \in M$ that is visited infinitely often by π. Thus, a tail of π can be seen as consecutive cycles over x. Since all cycles that pass through M are Odd winning (from item 2), we conclude that π is Odd winning.

Thm. 7 now follows as a corollary of Prop. 14.

5 Zielonka's Algorithm for Odd-Fair Parity Games

In this section, we construct a Zielonka-like algorithm that solves Odd-fair parity games. We call this algorithm *Odd-fair Zielonka's algorithm*. We first recall Zielonka's original algorithm in Sec. 5.1 and outline the changes imposed for our new Odd-fair version in Sec. 5.2. We then discuss the correctness of this new algorithm in Sec. 5.4.

From now on we take $\mathcal{G}^\ell = \langle (V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi), E^\ell \rangle$ to be an Odd-fair parity game.

5.1 Zielonka's Original Algorithm

Intuitively, Zielonka's algorithm consists of two nested recursive functions, $\mathsf{SOLVE}_{\mathsf{Even}}(n, \mathcal{G})$ and $\mathsf{SOLVE}_{\mathsf{Odd}}(n, \mathcal{G})$ which compute $\mathcal{W}_{\mathsf{Even}}$ and $\mathcal{W}_{\mathsf{Odd}}$ in a given parity game $\mathcal{G}$ with, respectively, even or odd upper bound priority n. Both functions recursively call each other on a sequence of sub-games that is constructed during the run of the algorithm.

The main difference between Zielonka's original algorithm [46] and our new Odd-fair version in Alg. 1 is the computation of the safe reachability set, denoted by $\mathsf{SafeReach}^f_\Lambda$ within the algorithms. Intuitively, the safe reachability set of player Λ is the set of vertices from which Λ has a strategy to force the game into the reach set $R \subseteq V$, while staying in the safety set $S \subseteq V$. In a (normal) parity game $\mathcal{G}$ (without live edges), this set can be computed via the single-nested fixed-point formula

$$\mathcal{X}_\Lambda := \mu X \ . \ (S \cap (R \cup \mathsf{Cpre}_\Lambda(X))). \tag{10}$$

If one interpretes Alg. 1 over (normal) parity games $\mathcal{G}$, defines $\mathsf{SafeReach}^f_\Lambda$ via (10) for the respective player, and replaces $\mathsf{SafeReach}^f_{\mathsf{Odd}}(\cdot, X, \cdot)$ in the last return statement with X (so, the algorithm returns X for any Λ), one gets exactly Zielonka's algorithm for parity games.

Algorithm 1 Odd-Fair Zielonka's Algo.

```
procedure SOLVE_Λ(n, G^ℓ)
    X ← V
    Z_¬Λ ← G
    while Z_¬Λ ≠ ∅ do
        N ← {v | v ∈ X with χ(v) = n}
        Z ← X \ SafeReach^f_Λ(X, N, G^ℓ)
        Z_¬Λ ← SOLVE_¬Λ(n − 1, G^ℓ[Z])
        X ← X \ SafeReach^f_¬Λ(X, Z_¬Λ, G^ℓ)
    end while
    if Λ = Even then return X
    else return SafeReach^f_Odd(V, X, G^ℓ)
    end if
end procedure
```

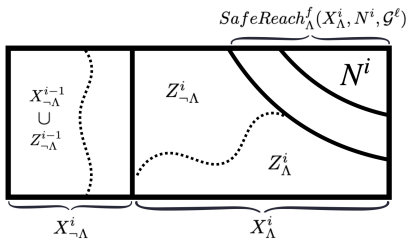


Figure 3 Visualization of the sets in Alg. 1.

5.2 The Odd-fair Zielonka's Algorithm

We are now considering an Odd-fair parity game $\mathcal{G}^\ell$. As discussed before, the main difference of the Odd-fair Zielonka's algorithm from the original one lies in the construction of the safe reachability sets denoted by $\mathsf{SafeReach}^f_\Lambda$ in Alg. 1. We therefore start by discussing its computation for both players.

The Odd Player. The first, somehow surprising, observation is that for player Odd in Odd-fair parity game $\mathcal{G}^\ell$, the safe reachability set $\mathcal{X}_{\mathsf{Odd}}$ can still be computed via (10). This is due to the fact that R only needs to be visited once, and Even vertices do not have live outgoing edges that might prevent player Odd from forcing a visit to R.

In addition, we can extract a *partial strategy template* for player Odd from the iterative computation of (10) via a similar, but much simpler ranking argument as used in Sec. 4. Here, $\mathsf{rank}(v) = 1$ for $v \in R$ and for the remaining vertices, $\mathsf{rank}(v)$ is the minimum integer j for which $v \in X^j := (S \cap (R \cup \mathsf{Cpre}_{\mathsf{Odd}}(X^{j-1})))$ where $X^0 = \emptyset$. The positional strategy of Λ is then to take the minimum ranked successor from each Odd node.

Another way to think about this strategy is in the form of an acyclic subgraph of $\mathcal{G}^\ell$ on $\mathcal{X}_{\mathsf{Odd}}$, where nodes in R have no outgoing edges, and for the remaining nodes, Odd nodes have one outgoing edge and Even nodes have all their outgoing edges. This is because if $v \in X^j \cap V_{\mathsf{Even}}$, all outgoing edges achieve positive progress towards R, i.e. for all $(v, w) \in E$, $w \in X^{j-1}$. Now it is easy to see that this subgraph almost defines a strategy template, i.e., on $\mathcal{X}_{\mathsf{Odd}} \setminus R$, Even nodes have all their outgoing edges in the subgraph, no Odd node lies on a cycle and all of them have one outgoing edge. However, vertices in R are dead-ends. We therefore call the strategy template induced by (10) *partial* and denote it by sr.

The Even Player. It follows from the results of Banerjee et. al. [5] that the safe reachability set $\mathcal{X}_{\mathsf{Even}}$ of player Even in Odd-fair parity games requires the 2-nested fixed-point formula $\nu Y.\mu X.S \cap (R \cup \mathsf{Apre}(Y, X))$, which (via the operators defined in Sec. 4.1) equals

$$\mathcal{X}_{\mathsf{Even}} := \nu Y \,.\, \mu X \,.\, S \cap (R \cup (\mathsf{Cpre}_{\mathsf{Even}}(X) \cup (\mathsf{Lpre}^{\exists}(X) \cap \mathsf{Pre}^{\forall}_{\mathsf{Odd}}(Y)))) \tag{11}$$

Intuitively, the necessity of a 2-nested formula arises from the following lack of information: we do not know in advance, which Odd nodes need to lie on a cycle on a strategy template required for Odd to win. If any positional strategy that lets Odd win (i.e., to avoid R or leave S) from a $v \in V^\ell$ requires v to lie on a cycle, then Odd has to take v's live outgoing edges as well, and thus, it can enter $\mathcal{X}_{\mathsf{Even}}$ and lose. The calculation of (11) starts with $Y^0 := V$, resulting in $\mathsf{Pre}^{\forall}_{\mathsf{Odd}}(V) = V$, hence

$$Y^1 := \mu X \ . \ S \cap (R \cup \mathsf{Cpre}_{\mathsf{Even}}(X) \cup \mathsf{Lpre}^{\exists}(X)). \tag{12}$$

Due to the disappearence of $\mathsf{Pre}^{\forall}_{\mathsf{Odd}}(Y)$ in this iteration, intuitively all $v \in V^\ell$ are treated as if they do not have any positional winning Odd strategy on them, so as if all Odd strategies have to take all the live edges in the game. Y^1 includes any Odd vertex that progresses towards R while staying in S with using either all its edges (due to $\mathsf{Cpre}_{\mathsf{Even}}(X)$) or through one live edge (due to $\mathsf{Lpre}^{\exists}(X)$). Thus, any vertex that manages to stay in $V \setminus Y^1$ does so due to being won by Odd even if Even could force all the live outgoing edges to be taken. Note that due to the monotonicity of fixed-point operators, for all j, $V \setminus Y^1 \subseteq V \setminus Y^j$.

Throughout the calculation, $V \setminus Y^j$ keeps track of the nodes that have managed to escape S or avoid R in the previous iteration, so are "already" won by Odd in the first j iterations. The inner fixed-point calculation in the $(j+1)^{th}$ iteration treats $V \setminus Y^j$ as a subset of Odd's winning region and it deems any node that can be forced by Odd to reach $V \setminus Y^j$, lost by Even. When the algorithm saturates, Y^∞ contains only those Odd nodes that cannot be forced by Odd to reach $V \setminus Y^\infty$, i.e., are won by Even. Here it is important to observe that, $V \setminus Y^\infty$ contains some Odd nodes that are not $V \setminus Y^1$. Since they are in Y^1, these nodes inductively reach Even winning vertices through live edges. This reveals that, all nodes in $V \setminus Y^j$ but not in $V \setminus Y^1$ win due to a positional Odd strategy that reaches $V \setminus Y^{j-1}$. Iteratively, this reveals that all such nodes have positional Odd strategies that make them reach $V \setminus Y^1$.

The above alternative interpretation of the computation of $\mathcal{X}_{\mathsf{Even}}$ in (11) is the key insight that we utilize to define our new Odd-fair Zielonka's algorithm, as discussed next.

The Odd-fair Zielonka's Algorithm. Following up on the previous discussion, we use the following insight within the construction of the Odd-fair Zielonka's algorithm. We assume the existence of a core subset $\mathcal{W}'_{\mathsf{Odd}} \subseteq \mathcal{W}_{\mathsf{Odd}}$ that player Odd can force all nodes in $\mathcal{W}_{\mathsf{Odd}}$ to, that is winning for Odd even under the assumption that Even can force all the live edges in the game to be taken. Since Zielonka's algorithm solves parity games by a sequence of nested safe-reachability calculations for alternating players, we apply the following trick: Instead of computing $\mathcal{X}_{\mathsf{Even}}$ via (11) in each recursive call of Alg. 1, we only compute Y^1 via (12) and use it as an *overapproximation* of $\mathcal{X}_{\mathsf{Even}}$ (which is indeed the case due to the monotonicity of (11) in Y). That is, while we take the Odd safe reachability set $\mathsf{SafeReach}^f_{\mathsf{Odd}}$ as the original (linear) Odd safe reachability computation known for these games (given in (10)), we do not take Even safe reachability formula $\mathsf{SafeReach}^f_{\mathsf{Even}}$ to be the (quadratic) Even safe reachability computation known for these games (given in (11)), but we instead take it as its (linear) subformula given in (12) and arrive at an overapproximation of the Even safe reachability region at the end of each $\mathsf{SafeReach}^f_{\mathsf{Even}}$ calculation. We finalize the recursive call $\mathsf{SOLVE}_{\mathsf{Odd}}$ by an extra call of $\mathsf{SafeReach}^f_{\mathsf{Odd}}$ applied to the (thus) underapproximated Odd winning region in the sub-game, therefore expanding the returned Odd winning region of the sub-game.

By this, it turns out that the recursive call of $\mathsf{SOLVE}_{\mathsf{Odd}}(n, \mathcal{G}^\ell)$ actually computes $\mathcal{W}'_{\mathsf{Odd}}$ as the set X and we ensure that $\mathcal{W}_{\mathsf{Odd}}$ is returned by the additional (linear) computation of $\mathsf{SafeReach}^f_{\mathsf{Odd}}$ over X in the last return statement of Alg. 1. This instantiation of the safe-reachability computations is formalized next.

▶ **Definition 15.** *Given an Odd-fair parity game $\mathcal{G}^\ell = \langle (V, V_{Even}, V_{Odd}, E, \chi), E^\ell \rangle$ the safe-reachability procedures $\mathsf{SafeReach}^f_{Odd}(S, R, \mathcal{G}^\ell)$ and $\mathsf{SafeReach}^f_{Even}(S, R, \mathcal{G}^\ell)$ in Alg. 1 denote the iterative fixed-point computations in* (10) *for Odd and* (12) *for Even.*

5.3 Complexity of the Odd-fair Zielonka's Algorithm

The safe-reachability computations defined in Def. 15 have the same complexity as their computations via (10) in Zielonka's original algorithm. The only difference is in the number of calculated Pre operations: while $\mathsf{SafeReach}_{\mathsf{Even}}$ from Zielonka's original algorithm (10) require the calculation of only one Pre operator, $\mathsf{SafeReach}^{f}_{\mathsf{Even}}$ from (12) requires the calculation of 2 Pre operators. The additional final call of $\mathsf{SafeReach}^{f}_{\mathsf{Odd}}$ in $\mathsf{SOLVE}_{\mathsf{Odd}}$ procedure also has linear complexity and requires one Pre calculation. Therefore, not only the worst-case time complexity of Alg. 1 is equivalent to that of Zielonka's original algorithm (which would be the case even if we used the quadratic safe reachability formula from (11) for Even since the overall complexity of the algorithm is exponential) but we create almost no additional computational overhead in the algorithm by introducing the fairness assumptions.

We further remark that Alg. 1 is not a straight-forward interpretation of the nested fixed-point in (7), and its negation (see (14) in App. A.1 of [37]) in the form of Zielonka's algorithm. Firstly, such a straightforward approach is non-trivial due to Apre and Npre operators taking two variables from two different iterations of the fixed-point calculation. Furthermore, at each Even safe-reachability call of Alg. 1, as mentioned we compute 2 Pre operators (equation 12), whereas in each such corresponding step in the fixed-point iteration, we would have to compute 3 Pre operators due to the expansion of Apre (5b) and Npre (5c).

It remains to show that Odd-fair Zielonka's algorithm solves Odd-fair parity games.

5.4 Correctness of the Odd-fair Zielonka's Algorithm

We first recall that Odd-fair parity games are determined. Next, we prove the correctness of the algorithm by induction on n. Since in the base case $n = 0$ the calls correctly return $\emptyset$, it suffices to prove the correctness of each function, assuming the correctness of the other. This is formalized next.

▶ **Theorem 16** (Correctness of SOLVE_Λ, Alg. 1)**.** *Assume that for any Odd-fair parity game $\mathcal{G}'^{\ell}$ where $n' < n$ is an odd (resp. even) upper bound on the priorities of the game, $SOLVE_{\mathsf{Odd}}(n', \mathcal{G}')^{\ell}$ correctly returns the Odd winning region (resp. $\mathsf{SOLVE}_{\mathsf{Even}}(n', \mathcal{G}')^{\ell}$ correctly returns the Even winning region) in $\mathcal{G}'^{\ell}$. Then $SOLVE_\Lambda(n, \mathcal{G}^{\ell})$ correctly returns the winning region of player Λ where n is even if $\Lambda = \mathsf{Even}$ and odd if $\Lambda = \mathsf{Odd}$.*

Notation. We follow the notation of Küsters' proof [28] of Zielonka's original algorithm [46]. Recall that $\mathcal{G}^{\ell}$ has no dead-ends. For some $X \subseteq V$, we call $\mathcal{G}^{\ell}[X] = \langle (X, X \cap V_{\mathsf{Even}}, X \cap V_{\mathsf{Odd}}, X \times X \subseteq E, \chi\,|_X), X \times X \subseteq E^{\ell} \rangle$ a *subgame* of $\mathcal{G}^{\ell}$ if it has no dead-ends. Here, $\chi\,|_X$ is the priority function $\chi : V \to \mathbb{N}$ restricted to domain X. Let n be an upper bound on the priorities in V. If the parity of n is even, set Λ to Even; if it's odd, set Λ to Odd.

Λ-trap and Λ-paradise. A Λ-trap is a subset $T \subseteq V$ for $\Lambda \in \{\mathsf{Even}, \mathsf{Odd}\}$ such that, $\forall v \in T \cap V_{\neg\Lambda},\ \exists (v, w) \in E$ with $w \in T$ and $\forall v \in T \cap V_{\Lambda},\ (v, w) \in E \implies w \in T$. A Λ-paradise in $\mathcal{G}^{\ell}$ is a subset $T \subseteq V$ which is a $\neg\Lambda$-trap in V and there exists a winning Λ strategy template (T, E') in $\mathcal{G}^{\ell}$.

The recursive calls of SOLVE_Λ and $\mathsf{SOLVE}_{\neg\Lambda}$ on subgames within Alg. 1 induce a characteristic partition of the game graph. For the correctness proof, we need to remember a series of these subgames that are constructed through previous recursive calls. The partition of these subsets is illustrated in Fig. 3 and formalized as follows.

$$X^i_\Lambda := V \setminus X^i_{\neg\Lambda} \qquad N^i := \{v \in X^i_\Lambda \mid \chi(v) = n\} \tag{13}$$
$$Z^i := X^i_\Lambda \setminus \mathsf{SafeReach}^f_\Lambda(X^i_\Lambda, N^i, \mathcal{G}^\ell) \qquad X^{i+1}_{\neg\Lambda} := \mathsf{SafeReach}^f_{\neg\Lambda}(V, X^i_{\neg\Lambda} \cup Z^i_{\neg\Lambda}, \mathcal{G}^\ell)$$

here, in addition Z^i_Λ is the Λ winning region in the subgame $\mathcal{G}^\ell[Z^i]$. Intuitively, the sets constructed in (13) correspond to the sets with the same name within Alg. 1.

We collect the following observations on these sets, which are proven in [37].

1. **([37] – Obs. 37)** $X^i_{\neg\Lambda}$ is an Λ-trap, X^i_Λ, Z^i and Z^i_Λ are $\neg\Lambda$-traps in V. Z^i is in $\neg\Lambda$-trap in X_Λ and $Z^i_{\neg\Lambda}, Z^i_\Lambda$ are Λ- and $\neg\Lambda$-traps in Z^i, respectively. Therefore, $\mathcal{G}^\ell[Y]$ is a subgame of $\mathcal{G}^\ell$ with Y being any of these sets.
2. **([37] – Lem. 38)** $X^i_{\neg\Lambda} \cup \mathsf{SafeReach}^f_{\neg\Lambda}(X^i_\Lambda, Z^i_{\neg\Lambda}, \mathcal{G}^\ell) = \mathsf{SafeReach}^f_{\neg\Lambda}(V, X^i_{\neg\Lambda} \cup Z^i_{\neg\Lambda}, \mathcal{G}^\ell)$.
3. **([37] – Cor. 39)** As a consequence of the previous item, $\{X^i_{\neg\Lambda}\}_{i\in\mathbb{N}}$ is an increasing sequence. Consequently, $\{X^i_\Lambda\}_{i\in\mathbb{N}}$ is a decreasing sequence. As V is finite, this immediately implies that these sequences reach a saturation value for some, and in fact the same, k.
4. **([37] – Lem. 34)** If $R \subseteq V$ is an Odd-paradise in $\mathcal{G}^\ell$, then $\mathsf{SafeReach}^f_{\mathsf{Odd}}(V, R, \mathcal{G}^\ell)$ is also an Odd-paradise in $\mathcal{G}^\ell$.
5. **([37] – Lem. 31)** The set $U \setminus \mathsf{SafeReach}_\Lambda(U, R, \mathcal{G}^\ell)$ is a Λ-trap in U.

In contrast to Zielonka's original algorithm, the proof of the procedures $\mathsf{SOLVE}_{\mathsf{Even}}$ and $\mathsf{SOLVE}_{\mathsf{Odd}}$ is not identical in Odd-fair Zielonka's algorithm. This is due to the different safe-reachability set constructions used. Next we sketch the correctness proof of Thm. 16 for $\Lambda := \mathsf{Odd}$, corresponding to the correctness of procedure $\mathsf{SOLVE}_{\mathsf{Odd}}$. The proof for $\Lambda := \mathsf{Even}$ is left to the appendix, as it resembles the proof Zielonka's original algorithm more.

▶ **Proposition 17.** *Given the premisses of Thm. 16 for $\Lambda = \mathsf{Odd}$, if $Z^k_{\mathsf{Even}} = \emptyset$ then $\mathsf{SafeReach}^f_{\mathsf{Odd}}(V, X^k_{\mathsf{Odd}}, \mathcal{G}^\ell)$ is an Odd-paradise and $V \setminus \mathsf{SafeReach}^f_{\mathsf{Odd}}(V, X^k_{\mathsf{Odd}}, \mathcal{G}^\ell)$ is an Even-paradise in $\mathcal{G}^\ell$.*

Within Prop. 17, the fact that $Z^k_{\mathsf{Even}} = \emptyset$ refers to the termination of the recursive call in Alg. 1 which results in the saturation of the sequence $\{X^i_{\mathsf{Odd}}\}_{i\in\mathbb{N}}$ with X^k_{Odd}. This implies that $\mathsf{SOLVE}_{\mathsf{Odd}}$ returns $T := \mathsf{SafeReach}^f_{\mathsf{Odd}}(V, X^k_{\mathsf{Odd}}, \mathcal{G}^\ell)$, which is an Odd-paradise and $V \setminus T$ an Even-paradise. With this, Thm. 16 follows from Prop. 17 for $\Lambda = \mathsf{Odd}$. We now give a proof sketch of Prop. 17.

We first recall from observation 1 that T and $V \setminus T$ are Even- and Odd-traps in V, respectively. In order to prove Prop. 17, it remains to show that there exists an Odd (resp. Even) strategy template which is winning in $\mathcal{G}^\ell$ and maximal on T (resp. $V \setminus T$). We next give the construction of these templates and a high-level intuition on why they are actually *winning*.

Winning Odd Strategy Templates. As X^k_{Odd} is known to be an Even-trap, it can be proven to be an Odd-paradise by constructing a winning maximal strategy template on it. It then follows from observation 4 that T is also an Odd-paradise.

Towards a construction of a maximal winning Odd strategy template on X_{Odd}, we first observe that $X^k_{\mathsf{Odd}} = Z^k_{\mathsf{Odd}} \cup \mathsf{SafeReach}^f_{\mathsf{Odd}}(X^k_{\mathsf{Odd}}, N^k, \mathcal{G}^\ell)$ (as $Z^k_{\mathsf{Even}} = \emptyset$). Then there exists a maximal winning Odd strategy template z on $Z^k = Z^k_{\mathsf{Odd}}$ in game $\mathcal{G}^\ell[Z^k]$. Any play π compliant with z that starts and stays in Z^k is clearly Odd winning. However, z is not necessarily an Odd strategy template in $\mathcal{G}^\ell$ since there are possibly some $(v, w) \in E$ with $v \in Z^k \cap V_{\mathsf{Even}}$ and $w \notin Z^k$. For all such edges, $w \in \mathsf{SafeReach}^f_{\mathsf{Odd}}(X^k_{\mathsf{Odd}}, N^k, \mathcal{G}^\ell)$ since X^k_{Odd} is an Even-trap in V. For the state set $\mathcal{X}_{\mathsf{Odd}} := \mathsf{SafeReach}^f_{\mathsf{Odd}}(X^k_{\mathsf{Odd}}, N^k, \mathcal{G}^\ell)$, recall from Sec. 5.2 that there exists partial strategy template sr defined on $\mathcal{X}_{\mathsf{Odd}}$ with dead ends in N^k.

Using the templates z and sr, we can construct a maximal candidate Odd strategy template on X^k_{Odd}. Following the intuition behind the construction of $\mathcal{S}^{\mathcal{G}^\ell}$ in Def. 12, we first define a base subgraph (X^k_{Odd}, E') with $E' \subseteq E$ s.t. $(v, w) \in E$ is in E' if either (i) $(v, w) \in z \cup sr$, (ii) $v \in V_{\mathsf{Even}} \cap X^k_{\mathsf{Odd}}$, or (iii) $v \in N^k \cap V_{\mathsf{Odd}}$ and $w = v_r$ where v_r is a random fixed successor of v, that is in X^k_{Odd}. Such a successor is guaranteed to exist since X^k_{Odd} is an Even-trap. We now extend the subgraph (X^k_{Odd}, E') to an Odd strategy template by adding all live edges originating in vertices $X^k_{\mathsf{Odd}} \cap V^\ell$ that lie on a cycle in E', similar to Def. 12 (S3)-(S4). This results in a subgraph $\mathcal{S} = (X^k_{\mathsf{Odd}}, \overline{E'})$ that is a maximal Odd strategy template. The underlying idea behind $\mathcal{S}$ being winning is the following: Any play that starts in X^k_{Odd} either stays in Z^k after some point and is won by $\mathcal{S}$ collapsing to z, or sees a newly added cycle (one that is not in $z \cup sr$) infinitely often. All such cycles contain a newly added edge. An analysis of newly added edges reveal that, all of them – when seen infinitely often – eventually drag a play towards N^i. Thus, every play that sees a new cycle infinitely often sees n infinitely often, and thus won by Odd.

Winning Even Strategy Templates. Here we show that $V \setminus T$ is an Even-paradise in $\mathcal{G}^\ell$. We first define $\mathcal{X}^i_{\mathsf{Even}} := \mathsf{SafeReach}^f_{\mathsf{Even}}(X^i_{\mathsf{Odd}}, Z^i_{\mathsf{Even}}, \mathcal{G}^\ell)$ and denote by sr^i the partial Even strategy template defined on $\mathcal{X}^i_{\mathsf{Even}}$. We further denote the winning Even strategy on Z^i_{Even} in game $\mathcal{G}^\ell[Z^i]$ by z^i. We can now construct the Even strategy template $\mathcal{S} = (V \setminus T, E')$ where E' is the combination of edges in $sr^i \cup z^i$ with $\{(v, w) \in E \mid v \in V_{\mathsf{Odd}} \cap (V \setminus T)\}$. Since $V \setminus T$ is an Odd-trap by observation 5, the edge set E' stays within $V \setminus T$, i.e. $E' \subseteq V \setminus T \times V \setminus T$. Then clearly, $\mathcal{S}$ is an Even strategy template. To see $\mathcal{S}$ is winning we first observe that each $v \in V \setminus T$ there exists a unique $i < k$ such that $v \in \mathcal{X}^i_{\mathsf{Even}}$. Let $\pi = v_1 v_2 \ldots$ be a play compliant with $\mathcal{S}$ and let $s = \mathcal{X}_1 \mathcal{X}_2 \ldots$ be the sequence such that $v_i \in \mathcal{X}$. (1) If $v_j \in Z^i_{\mathsf{Even}}$, $v_{j+1} \in Z^i_{\mathsf{Even}} \cup \{\mathcal{X}^r_{\mathsf{Even}} \mid r < i\}$. This follows from Z^i_{Even} being an Odd-trap in X^i_{Odd}. (2) If π visits $v \in \mathcal{X}^i$ infinitely often, π visits Z^i_{Even} infinitely often: This is because π visits the (v, w) in $\mathcal{S}$ that makes positive progress towards Z^i_{Even} infinitely often as well. Let i be the minimum index such that $\mathcal{X}^i_{\mathsf{Even}}$ is seen infinitely often in s. By (1), π visits Z^i_{Even} infinitely often and by (1) and the minimality of i, it should eventually stay in Z^i_{Even}. Thus $\mathcal{S}$ eventually collapses to z^i_{Even} on π and the play is won by Even.

5.5 Experimental Results

We conducted an experimental study to empirically validate the claim that our new Odd-fair Zielonka's algorithm retains its efficiency in practice (see App. A.2 for details).

We generated Odd-fair parity instances manipulating 286 benchmark instances of the dataset PGAME_Synth_2021 of the SYNTCOMP benchmark suite [18] and 51 instances of PGSolver dataset of Keiren's benchmark suite [21] by adding live edges to the given (normal) parity games. We empirically compared the (non-optimized[6]) C++-based implementations of (i) the Odd-fair Zielonka's algorithm (`OF-ZL`) from Alg. 1, (ii) the "normal" Zielonka's algorithm (`N-ZL`) from [46], (iii) the fixed-point algorithm for Odd-fair parity games (`OF-FP`) from [5] implementing (7), and (iv) the "normal" fixed-point algorithm (`N-FP`) for "normal" parity games from [13]. On the *SYNTCOMP benchmarks*, the time-out rates are: 82 instances for `OF-FP`, 58 for `OF-ZL`; 73 for `N-FP` and 47 for `N-ZL`. On the 204 instances that neither of the algorithms time out the average computation times are: 122.7 seconds for `OF-FP`, 4.6

[6] While optimized version of `N-ZL` and `N-FP` are available in `oink` [44] our goal is a conceptual comparison, which is better achieved by similar (non-optimized) implementations for all algorithms.

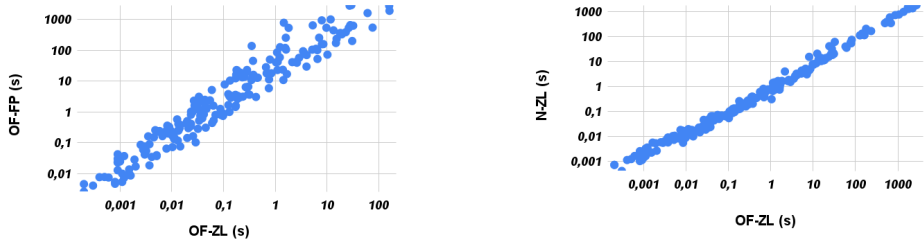


Figure 4 Scatter plot for a comparative evaluation of `OF-ZL` vs. `OF-FP` (left), and `OF-ZL` vs. `N-ZL` (right). Both plots show computation times in seconds using logarithmic scaling.

seconds for `OF-ZL`, 45.2 seconds for `N-FP` and 3.6 seconds for `N-ZL`. For all instances that did not time out for all four algorithms, Fig. 4 shows scatter plots comparing the computation times of `OF-ZL` with `OF-FP` (left) and `OF-ZL` with `N-ZL` (right) using logarithmic scaling. The diagonal shows instances with similar computation times. Points above the diagonal show superior performance of `OF-ZL`. For the *PGSolver dataset* `OF-FP` timed out on all generated instances, whereas `OF-ZL` took 24.9 seconds on average to terminate.

We clearly see that `OF-ZL` performs up to one order of magnitude better than `OF-FP` in many instances while `OF-ZL` and `N-ZL` perform very similar on the given benchmark instances. In addition, we observe that `OF-FP` starts timing out as soon as the examples became more complex. These outcomes match the known comparison results between the naive fixed-point calculation versus Zielonka's algorithm, on normal parity games.

References

1. Rajeev Alur, Salar Moarref, and Ufuk Topcu. Counter-strategy guided refinement of GR(1) temporal logic specifications. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, pages 26–33. IEEE, 2013.
2. Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin. Stochastic fairness and language-theoretic fairness in planning in nondeterministic domains. In J. Christopher Beck, Olivier Buffet, Jörg Hoffmann, Erez Karpas, and Shirin Sohrabi, editors, *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pages 20–28. AAAI Press, 2020.
3. André Arnold, Damian Niwiński, and Paweł Parys. A quasi-polynomial black-box algorithm for fixed point evaluation. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
4. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
5. Tamajit Banerjee, Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Fast symbolic algorithms for omega-regular games under strong transition fairness. *TheoretiCS*, 2, 2023.
6. Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 15. Springer, 2017.
7. Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012.
8. Krishnendu Chatterjee, Luca de Alfaro, Marco Faella, Rupak Majumdar, and Vishwanath Raman. Code aware resource management. *Formal Methods Syst. Des.*, 42(2):146–174, 2013.
9. Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*, 147(1-2):35–84, 2003.

10 Marco Daniele, Paolo Traverso, and Moshe Y. Vardi. Strong cyclic planning revisited. In Susanne Biundo and Maria Fox, editors, *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, pages 35–48. Springer, 1999.

11 Nicolás D'Ippolito, Natalia Rodríguez, and Sebastian Sardiña. Fully observable non-deterministic planning as assumption-based reactive synthesis. *J. Artif. Intell. Res.*, 61:593–621, 2018.

12 E. Allen Emerson and Charanjit S. Jutla. On simultaneously determinizing and complementing omega-automata (extended abstract). In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 333–342. IEEE Computer Society, 1989.

13 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377. IEEE Computer Society, 1991.

14 E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. *SIAM J. Comput.*, 29(1):132–158, 1999.

15 Nissim Francez. *Fairness*. Springer-Verlag, Berlin, Heidelberg, 1986.

16 Yuri Gurevich and Leo Harrington. Trees, automata, and games. 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, May 5-7, 1982, San Francisco, California, USA*, pages 60–65. ACM, 1982.

17 Daniel Hausmann and Lutz Schröder. Quasipolynomial computation of nested fixpoints. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings, Part I*, volume 12651 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2021.

18 Swen Jacobs, Guillermo A. Pérez, Remco Abraham, Véronique Bruyère, Michaël Cadilhac, Maximilien Colange, Charly Delfosse, Tom van Dijk, Alexandre Duret-Lutz, Peter Faymonville, Bernd Finkbeiner, Ayrat Khalimov, Felix Klein, Michael Luttenberger, Klara J. Meyer, Thibaud Michaud, Adrien Pommellet, Florian Renkin, Philipp Schlehuber-Caissier, Mouhammad Sakr, Salomon Sickert, Gaëtan Staquet, Clement Tamines, Leander Tentrup, and Adam Walker. The reactive synthesis competition (SYNTCOMP): 2018-2021. *CoRR*, abs/2206.00251, 2022. doi:10.48550/arXiv.2206.00251.

19 Marcin Jurdzinski. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000.

20 Marcin Jurdzinski, Rémi Morvan, and K. S. Thejaswini. Universal algorithms for parity games and nested fixpoints. In Jean-François Raskin, Krishnendu Chatterjee, Laurent Doyen, and Rupak Majumdar, editors, *Principles of Systems Design – Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, volume 13660 of *Lecture Notes in Computer Science*, pages 252–271. Springer, 2022.

21 Jeroen J. A. Keiren. Benchmarks for parity games. In Mehdi Dastani and Marjan Sirjani, editors, *Fundamentals of Software Engineering – 6th International Conference, FSEN 2015 Tehran, Iran, April 22-24, 2015, Revised Selected Papers*, volume 9392 of *Lecture Notes in Computer Science*, pages 127–142. Springer, 2015.

22 Nils Klarlund. *Progress Measures and Finite Arguments for Infinite Computations*. PhD thesis, Cornell University, USA, 1990.

23 Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Ann. Pure Appl. Log.*, 69(2-3):243–268, 1994.

24 Nils Klarlund and Dexter Kozen. Rabin measures and their applications to fairness and automata theory. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991*, pages 256–265. IEEE Computer Society, 1991.

25 Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

26 Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Where's waldo? sensor-based temporal logic motion planning. In *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*, pages 3116–3121. IEEE, 2007.

27 Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robotics*, 25(6):1370–1381, 2009.

28 Ralf Küsters. Memoryless determinacy of parity games. In Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors, *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2001.

29 Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Symbolic control for stochastic systems via parity games. *CoRR*, abs/2101.00834, 2021.

30 Shahar Maoz and Jan Oliver Ringert. Synthesizing a lego forklift controller in GR(1): A case study. In Pavol Cerný, Viktor Kuncak, and Parthasarathy Madhusudan, editors, *Proceedings Fourth Workshop on Synthesis, SYNT 2015, San Francisco, CA, USA, 18th July 2015*, volume 202 of *EPTCS*, pages 58–72, 2015.

31 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

32 Petter Nilsson, Necmiye Ozay, and Jun Liu. Augmented finite transition systems as abstractions for control synthesis. *Discret. Event Dyn. Syst.*, 27(2):301–340, 2017.

33 Paweł Parys. Parity games: Zielonka's algorithm in quasi-polynomial time. 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 10:1–10:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

34 Marco Pistore and Paolo Traverso. Planning as model checking for extended goals in non-deterministic domains. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 479–486. Morgan Kaufmann, 2001.

35 Jean-Pierre Queille and Joseph Sifakis. Fairness and related properties in transition systems – A temporal logic to deal with fairness. *Acta Informatica*, 19:195–220, 1983.

36 Miquel Ramírez and Sebastian Sardiña. Directed fixed-point regression-based planning for non-deterministic domains. In Steve A. Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI, 2014.

37 Irmak Sağlam and Anne-Kathrin Schmuck. Solving odd-fair parity games, 2023. `arXiv:2307.13396`.

38 Sven Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 369–384. Springer, 2008.

39 Robert S. Streett and E. Allen Emerson. The propositional mu-calculus is elementary. In Jan Paredaens, editor, *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.

40 María Svorenová, Jan Kretínský, Martin Chmelik, Krishnendu Chatterjee, Ivana Cerná, and Calin Belta. Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games. In Antoine Girard and Sriram Sankaranarayanan, editors, *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015*, pages 259–268. ACM, 2015.

41 Paulo Tabuada. *Verification and Control of Hybrid Systems – A Symbolic Approach*. Springer, 2009.

42 John G Thistle and RP Malhamé. Control of ω-automata under state fairness assumptions. *Systems & control letters*, 33(4):265–274, 1998.

43 Tom van Dijk. Attracting tangles to solve parity games. In Hana Chockler and Georg Weissenbacher, editors, *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 II*, volume 10982 of *Lecture Notes in Computer Science*, pages 198–215. Springer, 2018.

44 Tom van Dijk. Oink: An implementation and evaluation of modern parity game solvers. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, volume 10805 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2018.

45 Kai Weng Wong, Rüdiger Ehlers, and Hadas Kress-Gazit. Resilient, provably-correct, and high-level robot behaviors. *IEEE Trans. Robotics*, 34(4):936–952, 2018.

46 Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.

A Appendix

A.1 Proof of Prop. 14

We will restate the fixed-point formula that calculates the Odd winning region and the main proposition for the sake of self-containment.

▶ **Proposition 7.** *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ it holds that $Z = \mathcal{W}_{\mathsf{Odd}}$, where*

$$Z := \mu Y_l.\ \nu X_{l-1}.\ \ldots \mu Y_2.\ \nu X_1.\ \bigcap_{j\in[\![2,l]\!]} \mathcal{B}_j[Y_j, X_{j-1}], \tag{14}$$
$$\text{where} \quad \mathcal{B}_j[\mathbf{Y},\mathbf{X}] := \left(\bigcup_{i\in[j+1,l]} C_i\right) \cup \left(\overline{C_j} \cap \mathsf{Npre}(\mathbf{Y},\mathbf{X})\right) \cup \left(C_j \cap \mathsf{Cpre}_{\mathsf{Odd}}(\mathbf{Y})\right).$$

then $Z = \mathcal{W}_{\mathsf{Odd}}$. Further, it takes $\mathcal{O}(n^{l+1})$ symbolic steps to compute Z.

▶ **Proposition 14.** *Every player Odd strategy compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ is winning for Odd in $\mathcal{G}^\ell$.*

The main observation behind the proof of Prop. 14 is similar to the main observation in Sec. 5, leading to the proof of Alg. 1. That is, there exists a core subset of the Odd winning region $\mathcal{W}'_{\mathsf{Odd}} \subseteq \mathcal{W}_{\mathsf{Odd}}$, that is added to Z in the first iteration of the fixed-point calculation in (7), to which each $v \in \mathcal{W}_{\mathsf{Odd}}$ can be made to reach by Odd. Here in particular, we show that any Odd strategy compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ reaches $\mathcal{W}'_{\mathsf{Odd}}$ (infinitely often) while obeying the fairness condition, and is thus winning for Odd.

The proof of Prop. 14 consists of 3 main propositions. Before we present them, we will gather some observations from the fixed-point formula (7) and present them as lemmas.

According to our previous definitions, $Y_m^{r_l,r_{l-1},\dots,r_m}$ denotes the value of Y_m variable after the r_m^{th} iteration on it, while Y_i, X_i variables for $i > m$ are in their $r_i + 1^{th}$ iterations. If we flatten this formula we get the following equality: $Y_m^{r_l,r_{l-1},\dots,r_m} =$

$$\nu X_{m-1} \dots \mu Y_2 \nu X_1. \bigcap_{j \in [\![m+2,l]\!]} \mathcal{B}_j[Y_j^{r_j}, X_{j-1}^{r_{j-1}}] \cap \mathcal{B}_m[Y_m^{r_m-1}, X_{m-1}] \cap \bigcap_{j \in [\![2,m-2]\!]} \mathcal{B}_j[Y_j, X_{j-1}]$$

Observe that when the fixed-point above is calculated, all X_j, Y_j values for $j < m$ will saturate at the same value, which is the final result of the computation. That is,

▶ **Lemma 18.**

$$Y_m^{r_l,\dots,r_m} = \bigcap_{j \in [\![m+2,l]\!]} \mathcal{B}_j[Y_j^{r_j}, X_{j-1}^{r_{j-1}}] \cap \mathcal{B}_m[Y_m^{r_m-1}, Y_m^{r_l,\dots,r_m}] \cap \bigcap_{j \in [\![2,m-2]\!]} \mathcal{B}_j[Y_m^{r_l,\dots,r_m}, Y_m^{r_l,\dots,r_m}]$$

▶ **Lemma 19.** *For all $v \in \mathcal{W}_{\mathsf{Odd}}$ with $\mathsf{rank}(v) = (r_l, 0, \dots, r_2, 0)$. Then,*

$$v \in \bigcap_{j \in [\![2,l]\!]} Y_j^{r_l-1,0,r_{l-2}-1,0,\dots,r_{j-2}-1,0,r_j}$$

This is similar to our previous observation. $\mathsf{rank}(v) = (r_l, 0, \dots, r_2, 0)$ implies v was added to the formula while Y_j variable was on it's r_j^{th} iteration for all $j \in [\![2, l]\!]$. Since $X_{j-1}^0 = V$, the iteration values of X variables can be safely ignored.

▶ **Lemma 20.** *if $v \in V_{\mathsf{Even}}$,* $\forall (v, w) \in E, \mathsf{rank}(v) \geq_{l+1-\chi(v)} \mathsf{rank}(w)$

if $v \in V_{\mathsf{Odd}}$, $\exists (v, w) \in E, \mathsf{rank}(v) \geq_{l+1-\chi(v)} \mathsf{rank}(w)$

where $\mathsf{rank}(v) \geq_b \mathsf{rank}(w)$ denotes the $\geq$ relation in the lexicographic ordering, restricted to the first b elements of the tuple. If $\chi(v)$ is even, the inequalities are strict.

Proof. Consider a v with $\chi(v) \in \{m-1, m\}$ for an even m and let $\mathsf{rank}(v) = (r_l, 0, \dots, r_2, 0)$. By Lem. 19, $v \in Y_m^{r_l-1,0,\dots,r_{m-2}-1,0,r_m}$. If we look at the flattening of this formula in Lem.18, v is in particular, inside the middle term of this formula. That is,
$v \in \mathcal{B}_m[Y_m^{r_l-1,\dots,r_m-1}, Y_m^{r_l-1,\dots,r_m}]$. If we go through the definition of this term we get,

$$(\bigcup_{i \in [m+1,l]} C_i) \cup (\overline{C_m} \cap \mathsf{Npre}(Y_m^{r_l-1,\dots,r_m-1}, Y_m^{r_l-1,0,\dots,r_m})) \cup (C_m \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{r_l-1,0,\dots,r_m-1}))$$

That gives us, if $\chi(v) = m$, $v \in \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{r_l-1,0,\dots,r_m-1})$
if $\chi(v) = m-1$, $v \in \mathsf{Npre}(Y_m^{r_l-1,0,\dots,r_m-1}, Y_m^{r_l-1,0,\dots,r_m})$

By the definition of Npre we get, if $\chi(v) = m-1$ then $v \in \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{r_l-1,0,\dots,r_m})$. Since odd indices get 0-ranks, the claim of the lemma follows from the definition of $\mathsf{Cpre}_{\mathsf{Odd}}$ together with the observation $\mathsf{rank}(v) \geq_{l+1-m} \mathsf{rank}(w) \Leftrightarrow \mathsf{rank}(v) \geq_{l+1-(m-1)} \mathsf{rank}(w)$. ◀

Now we are ready to introduce the first of our three main propositions:

▶ **Proposition 21.** *If $\mathcal{W}_{\mathsf{Odd}} \neq \emptyset$, there exists a non empty set $M := \{v \in \mathcal{W}_{\mathsf{Odd}} \mid \mathsf{rank}(v) = (1, 0, 1, 0, \dots, 1, 0)\}$. Furthermore, for all $v \in M$, $\chi(v)$ is odd.*

Observe that $(1, 0, 1, 0, \ldots, 1, 0)$ is the smallest rank possible. Therefore, $v \in M$ are the vertices that were added to Z in (7) in the first iteration of the fixed-point calculation and were never removed. The first part of the proposition follows from the monotonicity of fixed-point calculation. That is, if M was empty Z would be empty as well.

For the second part, observe that in the first iteration of the formula, for all j, $Y_j = \emptyset$. Also, $\mathsf{Cpre}_{\mathsf{Odd}}(\emptyset) = \emptyset$. Then from (7), Z does not contain any v with even priority.

▶ **Proposition 22.** *All cycles in $\mathcal{S}^{\mathcal{G}^\ell}$ that pass through a vertex in M are Odd winning.*

To see why Prop. 22 holds, we make an observation. For an even $m \leq l$, let $Y_m^{\mathbb{I}}$ denote the value of Y_m after the first ever iteration over it is completed, during the computation of 7. I.e. $Y_m^{\mathbb{I}} = Y^{0,0,\ldots,0,1}$. Since for all j, $Y_j^0 = \emptyset$ and $X_{j-1}^0 = V$, Lem. 18 gives,

$$Y_m^{\mathbb{I}} = \bigcap_{j \in [\![m+2,l]\!]} \mathcal{B}_j[\emptyset, V] \cap \mathcal{B}_m[\emptyset, Y_m^{\mathbb{I}}] \cap \bigcap_{j \in [\![2,m-2]\!]} \mathcal{B}_j[Y_m^{\mathbb{I}} Y_m^{\mathbb{I}}] \tag{15}$$

If we go through the definition of $\mathcal{B}_j$ we see that: the first term of this formula adds or deletes $v \in C_j$ with $j > m$. It adds all the ones with odd j and removes all the ones with even j.The last term adds and removes $v \in C_j$ for $j \leq m-2$. It adds the ones in $\mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{\mathbb{I}})$ and removes the ones that are not. The middle term eliminates C_m and all $v \in C_j \cap \neg\mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}})$ for $j < m$, and adds $v \in C_{m-1} \cap \mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}})$. If we go through the definition of Npre, we see that $\mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}}) = \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{\mathbb{I}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_m^{\mathbb{I}}))$. This gives,

$$v \in Y_m^{\mathbb{I}} \iff \chi(v) > m \text{ and is odd, or } \chi(v) < m \text{ and } v \in \mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}}) \tag{16}$$

Then for all $v \in M$, $v \in Y_m^{\mathbb{I}}$ for each even $m \leq l$. In particular, $v \in Y_n^{\mathbb{I}}$ where n is such that $\chi(v) = n-1$. It follows that $v \in \mathcal{B}_n[\emptyset, Y_n^{\mathbb{I}}]$. Then, $v \in \mathsf{Cpre}_{\mathsf{Odd}}(Y_n^{\mathbb{I}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_n^{\mathbb{I}}))$. Since all live outgoing edges of v are in $Y_n^{\mathbb{I}}$, for all (v, w) in $\mathcal{S}^{\mathcal{G}^\ell}$, $w \in Y_n^{\mathbb{I}}$.

By our previous observation w either has an odd priority larger than n, or is in $\mathsf{Cpre}_{\mathsf{Odd}}(Y_n^{\mathbb{I}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_n^{\mathbb{I}}))$. If $\chi(w) > n$ is odd, then $w \in Y_{\chi(w)+1}^{\mathbb{I}}$, and we repeat the same argument to conclude the highest priority seen is always odd.

▶ **Definition 23.** *We call a play $\pi = v_1 v_2 \ldots$ in $\mathcal{S}^{\mathcal{G}^\ell}$* minimal *if for all $v_i \in V_{\mathsf{Odd}}$, v_{i+1} is the minimum ranked successor of v_i. A minimal cycle is a section of a minimal play.*

▶ **Lemma 24.** *Every minimal play is Odd winning.*

A minimal play only sees minimal cycles. Let $\delta = w_1 w_2 \ldots w_1$ be such a cycle. δ cannot be an Even winning cycle: Assume $b := \max\{\chi(w) \mid w \in \delta\}$ is even. Let $w_i \in \delta$ have priority b. By Obs. 20, $\mathsf{rank}(w_i) >_{l+1-b} \mathsf{rank}(w_{i+1}) \geq_{l+1-\chi(w_{i+1})} \ldots \geq_{l+1-\chi(w_{i-1})} \mathsf{rank}(w_i)$. Since for all $w_j \in \delta$, $\chi(w_j) \leq b$, the inequality yields $\mathsf{rank}(w_i) >_{l+1-b} \mathsf{rank}(w_i)$, which is a contradiction.

▶ **Proposition 25.** *Any minimal play compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ visits M infinitely often.*

Let $\delta = w_1 w_2 \ldots w_1$ be a minimal cycle and w_k its vertex with maximum priority. We will show that $w_k \in M$. Since $\pi = \delta\delta\ldots$ is a minimal play, by Lemma. 24 we know $\chi(w_k)$ is odd. Furthermore, we have observed in 16 that $w_k \in Y_m^{\mathbb{I}}$ for all $m > \chi(w_k)$. If we can show that $w_k \in Y_m^{\mathbb{I}}$ also for $m < \chi(w_k)$, then we have $w_k \in M$. We will now show this.

Assume to the contrary that $w_k \notin M$ and let j be the largest non-trivial index of $\mathsf{rank}(w_k)$. That is $j < l$ is the largest even integer such that $w_k \notin Y_j^{\mathbb{I}}$. Let t be the value of this index, i.e. $w_k \in Y_j^{0,\ldots,0,t} \setminus Y_j^{0,\ldots,0,t-1}$. Let us denote $Y_j^{0,\ldots,0,t}$ by $Y_j^{\mathbf{t}}$ for short.

Since δ is minimal, Lem. 20 gives $\mathsf{rank}(w_i) \geq_{l+1-\chi(w_i)} \mathsf{rank}(w_{i+1})$ for all $w_i \in \delta$. Since $\chi(w_i) \leq \chi(w_k)$ for all i and $\chi(w_k) < j$; $\mathsf{rank}(w_i) \geq_{l+1-j} \mathsf{rank}(w_{i+1})$ for all $w_i \in \delta$. This implies $\mathsf{rank}(w) =_{l+1-j} \mathsf{rank}(w')$ for all $w, w' \in \delta$. It follows that for all $w \in \delta$, $w \in Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$.

Once more by Lem. 18 we get that for all $w \in \delta$,

$$w \in \mathcal{B}_j[Y_j^{\mathbf{t-1}}, Y_j^{\mathbf{t}}] = (\bigcup_{i \in [j+1,l]} C_i) \cup (\overline{C_j} \cap \mathsf{Npre}(Y_j^{\mathbf{t-1}}, Y_j^{\mathbf{t}}) \cup (C_j \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_j^{\mathbf{t-1}})))$$

Since $\chi(w) < j$, this implies

$$w \in \mathsf{Npre}(Y_j^{\mathbf{t-1}}, Y_j^{\mathbf{t}}) = \mathsf{Cpre}_{\mathsf{Odd}}(Y_j^{\mathbf{t}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_j^{\mathbf{t}}) \cap \mathsf{Pre}^{\exists}_{\mathsf{Odd}}(Y_j^{\mathbf{t-1}}))$$

Now consider the set $Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$, which is initially empty. Then the first term in δ that gets in $Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$ has to be in $\mathsf{Pre}^{\exists}_{\mathsf{Odd}}(Y_j^{\mathbf{t-1}})$. This contradicts our assumption that all $w_i \in Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$ and proves that $w_k \in M$. We are now ready to prove the main theorem.

Proof of Thm. 14. Let $\pi = v_0 v_1 \ldots$ be a play compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ with $v_0 \in \mathcal{W}_{\mathsf{Odd}}$. Since π is compliant with an Odd strategy template, it is a fair play. For a node $v \in \mathcal{W}_{\mathsf{Odd}}$, let $v_{\min}$ be the minimum ranked successor of v. Since π is fair, for all v that is visited infinitely often in π, $v_{\min}$ is visited infinitely often as well. This gives us an infinite subsequence of π that is minimal. Since all minimal plays visit M infinitely often (Prop. 25), π visists M infinitely often. Then there must exist an $x \in M$ that π visits infinitely often. Then a tail of π is consisted of consecutive cycles over x. Since all cycles that pass through M are Odd winning (Prop. 22), π is Odd-winning. ◀

A.2 Details on Experimental Results

We conducted an experimental study to empirically validate the claim that our new Odd-fair Zielonka's algorithm retains its efficiency in practice. For this, we implemented the following algorithms (non-optimized) in C++:

- `OF-ZL`: Odd-fair Zielonka's algorithm (Alg. 1),
- `N-ZL`: "normal" Zielonka's algorithm from [46] (i.e., Alg. 1 with the simplifications described in Sec. 5.1),
- `OF-FP`: the fixed-point algorithm for Odd-fair parity games implementing (7) ,
- `N-FP`: the fixed-point algorithm for "normal" parity games from [13].

Of course, for both `N-ZL` and `N-FP` there exist optimized implementations (e.g. `oink` [44]). However, the goal of this section is to show a conceptual comparison, rather than evaluating best computation times. We believe this is better achieved using similar (non-optimized) implementations for all algorithms. In particular, by our experiments we show:

1. `OF-ZL`: significantly outperforms `OF-FP` on almost all benchmarks (Fig. 5 (right))
2. the performance of `OF-ZL` and `N-ZL` on the given benchmark set is very similar (Fig. 8),
3. the comparative performance of `OF-ZL` and `N-ZL` w.r.t. their respective fixed-point versions `OF-FP` and `N-FP`), respectively, is very similar (see Fig. 7).

All experiments where run on a large benchmark suite explained in Sec. A.2.1. To perform our experiments we used a machine equipped with Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz and 8GB RAM. We declare a timeout when the calculation of an example exceeds 1 hour.

A.2.1 Benchmark

We generated Odd-fair parity game instances manipulating 286 benchmark instances of PGAME_Synth_2021 dataset of the SYNTCOMP benchmark suite [18] and 51 benchmark instances of the PGSolver dataset of Keiren's benchmark suite [21]. Within the latter, we restricted ourselves to instances with ≤ 5000 nodes. Both datasets contain examples of

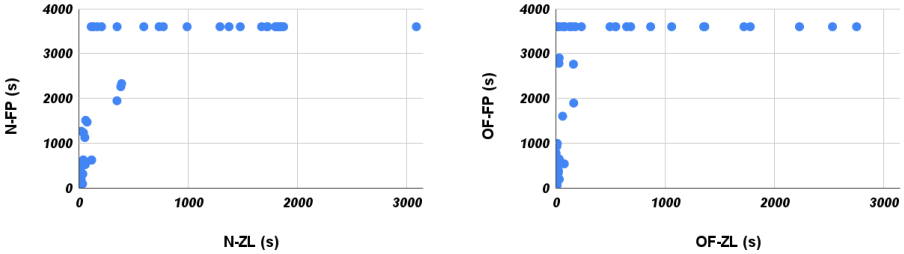


Figure 5 (Zoomed out version) A comparison of `N-FP` vs. `N-ZL` in regular parity games (left), and `OF-FP` vs. `OF-ZL` on fair parity games (right).

normal parity games. For each selected example, we generate Odd-fair parity game instances for a particular liveness percentage α. For a α%-liveness instant, we fix α% of the Odd nodes in the game, and turn α% of each of their outgoing edges to live edges.

On the Odd-fair instances with 50%−liveness generated from the SYNTCOMP benchmark suite, there are 204 instances where neither of the algorithms `OF-FP`, `OF-ZL`, `N-FP` or `N-ZL` timed out. On these instances, `OF-ZL` gives an average computation time of 4.6 seconds while `OF-FP` took 122.7 seconds on average. On the same examples, `N-ZL` takes on average 3.6 seconds to compute while `N-FP` takes 45.2 seconds. For the PGSolver dataset `OF-FP` timed out on all generated instances, whereas `OF-ZL` took 24.9 seconds on average to terminate.

A.2.2 Comparative Evaluation

In order to validate the computational advantage of `OF-ZL` over `OF-FP`, we ran both algorithms on all 50%-liveness instances generated from the SYNTCOMP benchmark dataset. On 58 of these instances, both algorithms time out. The run-times for all other instances are depicted in Fig. 5 (right), 6 (right) and 7 (right). The left plots in Fig. 5-7 show the same comparison for the "normal" parity algorithms `N-ZL` and `N-FP`. In both cases, Fig. 6 shows the zoomed-in version of the respective plot in Fig. 5. Fig. 7 shows the data-points from the respective plot in Fig. 6 as a scatter plot in log-scale. The examples on which only `x-FP` times out, can be seen as the dots on the ceiling of the plots in Fig. 5. In all plots, points above the diagonal correspond to instances where Zielonka's algorithm outperforms the fixed-point algorithm.

We clearly see in Fig. 5-7 that Zielonka's algorithm performs significantly better than the fixed-point version, both in the Odd-fair (right) and in the normal (left) case. More importantly, the overall performance comparison between `OF-ZL` over `OF-FP` (right plots) mimics the comparison between `N-ZL` over `N-FP`. This allows us to conclude that our new Odd-fair Zielonka's algorithm retains the computational advantages of Zielonka's algorithm.

A closer look at runtimes (see [37]) reveals `OF-ZL` has almost the same runtime as `N-ZL`, implying our changes in the algorithm incur almost no computational disadvantages over the original algorithm. This allows us to handle transition fairness for almost free in practice.

Conclusion. The results show that `OF-ZL` is significantly faster in solving Odd-fair parity games compared to `OF-FP`. The outcomes match the known comparison results between fixed-point calculation versus Zielonka's algorithm, on normal parity games.

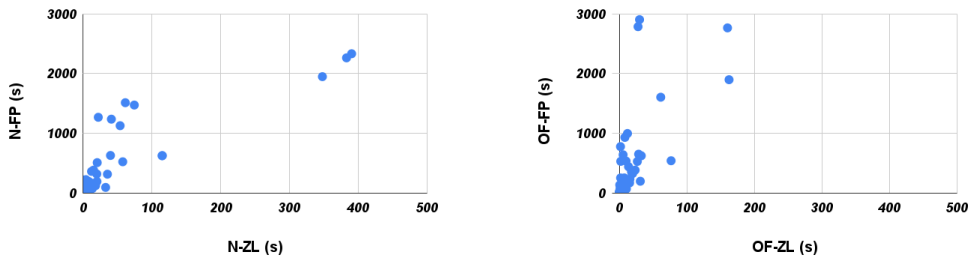


Figure 6 (Zoomed in version) A comparison of `N-FP` vs. `N-ZL` in regular parity games (left), and `OF-FP` vs. `OF-ZL` on fair parity games (right).

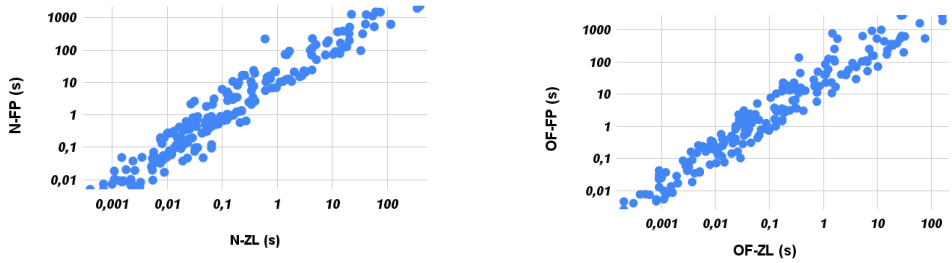


Figure 7 A comparison of `N-FP` vs. `N-ZL` in regular parity games (left), and `OF-FP` vs. `OF-ZL` on fair parity games (right) in terms of log-scale plots where the timeouts are removed.

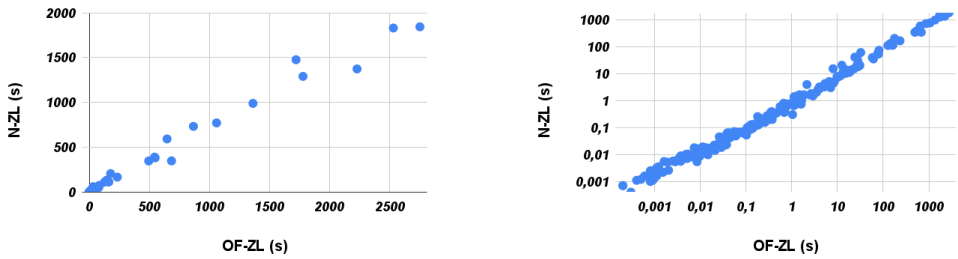


Figure 8 A comparison of `N-ZL` vs. `OF-ZL` over examples that do not timeout on both. Right hand side plot visualizes the same data in logscale.

New Lower Bounds for Reachability in Vector Addition Systems

Wojciech Czerwiński
University of Warsaw, Poland

Ismaël Jecker
University of Warsaw, Poland
FEMTO-ST, CNRS, Univ. Franche-Comté, France

Sławomir Lasota
University of Warsaw, Poland

Jérôme Leroux
LaBRI, CNRS, Univ. Bordeaux, France

Łukasz Orlikowski
University of Warsaw, Poland

Abstract

We investigate the dimension-parametric complexity of the reachability problem in vector addition systems with states (VASS) and its extension with pushdown stack (pushdown VASS). Up to now, the problem is known to be $\mathcal{F}_d$-hard for VASS of dimension $3d + 2$ (the complexity class $\mathcal{F}_d$ corresponds to the kth level of the fast-growing hierarchy), and no essentially better bound is known for pushdown VASS. We provide a new construction that improves the lower bound for VASS: $\mathcal{F}_d$-hardness in dimension $2d + 3$. Furthermore, building on our new insights we show a new lower bound for pushdown VASS: $\mathcal{F}_d$-hardness in dimension $\frac{d}{2} + 6$. This dimension-parametric lower bound is strictly stronger than the upper bound for VASS, which suggests that the (still unknown) complexity of the reachability problem in pushdown VASS is higher than in plain VASS (where it is Ackermann-complete).

2012 ACM Subject Classification Theory of computation → Concurrency; Theory of computation → Verification by model checking; Theory of computation → Logic and verification

Keywords and phrases vector addition systems, reachability problem, pushdown vector addition system, lower bounds

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.35

Funding *Wojciech Czerwiński, Ismaël Jecker and Łukasz Orlikowski*: Supported by the ERC grant INFSYS, agreement no. 950398.
Sławomir Lasota: Supported by the ERC grant INFSYS, agreement no. 950398 and by the NCN grant 2021/41/B/ST6/00535.
Jérôme Leroux: Supported by the grant ANR-17-CE40-0028 of the French National Research Agency ANR (project BRAVAS).

Acknowledgements We are grateful to Weijun Chen for his sharp observation that brought to light a technical error in the initial version of this paper.

1 Introduction

Petri nets, equivalently presentable as vector addition systems with states (VASS), are an established model of concurrency with widespread applications. The central algorithmic problem for this model is the *reachability problem* which asks whether from a given initial configuration there exists a sequence of valid execution steps reaching a given final configuration. For a long time the complexity of this problem remained one of the hardest open

© Wojciech Czerwiński, Ismaël Jecker, Sławomir Lasota, Jérôme Leroux, and Łukasz Orlikowski;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 35; pp. 35:1–35:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

questions in verification of concurrent systems. In 2019 Leroux and Schmitz made a significant breakthrough by providing an Ackermannian upper bound [14]. With respect to the hardness, the exponential space lower bound, shown by Lipton already in 1976 [16], remained the only known for over 40 years until a breakthrough non-elementary lower bound by Czerwiński, Lasota, Lazic, Leroux and Mazowiecki in 2019 [3, 4]. Finally, a matching Ackermannian lower bound announced in 2021 independently by two teams, namely Czerwiński and Orlikowski [5] and Leroux [12], established the exact complexity of the problem.

However, despite the fact that the exact complexity of the reachability problem for VASS is settled, there are still significant gaps in our understanding of the problem. One such gap is the complexity of the reachability problem *parametrised by the dimension*, namely deciding the reachability problem for d-dimensional VASS (d-VASS) for fixed $d \in \mathbb{N}$. Currently, the exact complexity bounds are only known for dimensions one and two. In these cases, the complexity depends on the representations of numbers in the transitions, either unary or binary. For binary VASS (where the numbers are represented in binary) the reachability problem is known to be NP-complete for 1-VASS [9] and PSpace-complete for 2-VASS [2]. For unary VASS the problem is NL-complete for both 1-VASS (folklore) and 2-VASS [8].

Much less is known for higher dimensions, and it is striking that even in the case of 3-VASS we have a huge complexity gap. The best complexity upper bound comes from the above mentioned work of Leroux and Schmitz [14], where it is proved that the reachability problem for $(d-4)$-VASS is in $\mathcal{F}_d$ (here $\mathcal{F}_d$ denotes the dth level of the Grzegorczyk hierarchy of complexity classes, which corresponds to the fast growing function hierarchy). In particular this shows that the reachability problem for 3-VASS is in $\mathcal{F}_7$ (recall that $\mathcal{F}_3 = \textsc{Tower}$).

The recent Ackermann-hardness results provide lower bounds for the reachability problem in fixed dimensions. The result of Czerwiński and Orlikowski [5] yields $\mathcal{F}_d$-hardness for $6d$-VASS, while the result of Leroux [12] establishes $\mathcal{F}_d$-hardness for $(4d+5)$-VASS. Lasota improved upon these results and showed $\mathcal{F}_d$-hardness of the problem for $(3d+2)$-VASS [10]. In [6], additional results were obtained for specific dimensions: PSpace-hardness for unary 5-VASS, ExpSpace-hardness for binary 6-VASS and Tower-hardness for unary 8-VASS.

To summarise, despite significant research efforts there are still several natural problems related to the VASS reachability problem that present significant complexity gaps:

$\mathbf{Q_1}$: What is the complexity of the reachability problem for VASS of dimension 3? It is known to be PSpace-hard and in $\mathcal{F}_7$;

$\mathbf{Q_2}$: What is highest dimension for which the complexity of the reachability problem is elementary? It is known to fall within the range of 2 to 8;

$\mathbf{Q_3}$: What is the smallest constant C such that the complexity of the reachability problem for d-VASS is in $\mathcal{F}_{Cd+o(d)}$? It is known to fall within the range of $\frac{1}{3}$ to 1.

In this work, we focus on addressing Question $\mathbf{Q_3}$. We present new and improved lower bounds, first in the standard setting of VASS, and then in the setting of pushdown VASS (PVASS) which extend the VASS model by incorporating a pushdown stack.

VASS reachability. Our first main result is a new complexity lower bound for the reachability problem which improves the gap:

▶ **Theorem 1.** *The reachability problem for $(2d+3)$-VASS is $\mathcal{F}_d$-hard.*

A preliminary version of this result was presented in [11]. In this revised version, we aim to present the result in a conceptually simple framework by using the notion of *triples*, a formalism that was originally developed in [3], and was heavily used in [5] and [10]. We view this contribution as an important step towards understanding the complexity of the VASS reachability problem parametrised by the dimension.

PVASS reachability. The decidability of the reachability problem for PVASS has been an important open problem for over a decade [1]. Despite efforts of the community, it remains unknown even for PVASS of dimension 1 (1-PVASS), namely automata with one counter and one pushdown stack.

▶ **Conjecture 2.** *The reachability problem for PVASS is decidable.*

It is important to acknowledge that the PVASS setting is complex, and few decidability results are known. Some progress has been made in the study of the *coverability* problem, a variant of the reachability problem which asks whether from a given initial configuration there exists a sequence of valid execution steps that reaches a configuration *greater* than the given final configuration. Notably, the coverability problem has been shown to be decidable for 1-PVASS [15] (and mentioned to be in ExpSpace).

Interestingly, despite the slow progress on determining upper bounds for PVASS, there is limited knownledge about lower bounds as well. To the best of our knowledge, the only lower bound that is not directly implied by the results on VASS concerns (again) the coverability problem for 1-PVASS, which has been established as PSpace-hard [7].

As for our contribution, our second main result is the first complexity lower bound for the PVASS reachability problem that is not immediately inherited from VASS:

▶ **Theorem 3.** *The reachability problem for* $(\lfloor \frac{d}{2} \rfloor + 6)$*-PVASS is* $\mathcal{F}_d$*-hard.*

Notably, Theorem 3 implies that for sufficiently large d the reachability problem for d-PVASS is harder than the problem for $(d+1)$-VASS (which is a subclass of d-PVASS as the pushdown stack can keep track of one VASS counter). Indeed the problem for d-PVASS is $\mathcal{F}_{2d-12}$-hard by Theorem 3, while the problem for $(d+1)$-VASS is in $\mathcal{F}_{d+5}$ by [14].

While our results indicate that the reachability problem for PVASS is harder than for VASS, some known results about PVASS hint that even higher lower bounds might be proved: In [13] it was shown that PVASS are able to weakly compute functions of hyper-Ackermannian growth rate. Based on this observation we propose the following two conjectures:

▶ **Conjecture 4.** *There exists a fixed dimension* $d \in \mathbb{N}$ *such that the reachability problem for* d*-PVASS is Ackermann-hard.*

▶ **Conjecture 5.** *The reachability problem for PVASS is hyperAckermann-hard.*

2 Preliminaries

Fast-growing hierarchy. Let $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$ be the set of positive integers. We define the complexity classes $\mathcal{F}_i$ corresponding to the ith level in the Grzegorczyk Hierarchy [18, Sect. 2.3, 4.1]. To this aim we choose to use the following family of functions $\mathbf{F}_i : \mathbb{N}_+ \to \mathbb{N}_+$, indexed by $i \in \mathbb{N}$:

$$\mathbf{F}_0(n) = n + 2, \qquad \mathbf{F}_{i+1} = \widetilde{\mathbf{F}_i} \quad \text{where} \quad \widetilde{F}(n) = F^{n-1}(1) = \underbrace{F \circ F \circ \ldots \circ F}_{n-1}(1). \tag{1}$$

In particular, $\mathbf{F}_1(n) = 2n - 1$ and $\mathbf{F}_i(1) = 1$ for all $i \in \mathbb{N}_+$. Using the functions $\mathbf{F}_i$, we define the complexity classes $\mathcal{F}_i$, indexed by $i \in \mathbb{N}_+$, of problems solvable in deterministic time $\mathbf{F}_i \circ \mathbf{F}_{i-1}^m(n)$ for some $m \in \mathbb{N}$:

$$\mathcal{F}_i = \bigcup_{m \in \mathbb{N}} \mathrm{DTime}(\mathbf{F}_i \circ \mathbf{F}_{i-1}^m(n)).$$

Intuitively speaking, the class $\mathcal{F}_i$ contains all problems solvable in time $\mathbf{F}_i(n)$, and is closed under reductions computable in time of lower order $\mathbf{F}_{i-1}^m(n)$, for some fixed $m \in \mathbb{N}_+$. In particular, $\mathcal{F}_3 = \textsc{Tower}$ (problems solvable in a tower of exponentials of time or space whose height is an elementary function of input size). The classes $\mathcal{F}_i$ are robust with respect to the choice of fast-growing function hierarchy (see [18, Sect.4.1]). For $i \geq 3$, instead of deterministic time, one could equivalently take nondeterministic time, or space as all these definitions collapse.

2.1 Counter programs

A *counter program* (or simply a *program*) is a sequence of (line-numbered) commands, each of which is of one of the following types:

$\mathsf{x} \mathrel{+}= 1$	(increment the counter x by one)
$\mathsf{x} \mathrel{-}= 1$	(decrement the counter x by one)
goto L **or** L'	(nondeterministically jump to either line L or line L')
zero? x	(zero test: continue if counter x equals 0)

Counter programs *with a pushdown stack* (or simply *programs with stack*) are enhanced versions of plain counter programs that incorporate a stack containing a word over a fixed stack alphabet $\mathbf{S}$. The stack content is modified by using two additional types of commands:

$\textsc{push}(s)$	(push the symbol $s \in \mathsf{S}$ at the top of the stack)
$\textsc{pop}(s)$	(pop the symbol $s \in \mathsf{S}$ if it is at the top of the stack)

The command $\textsc{pop}(s)$ fails if the stack is empty or if the top symbol is different from s. A *configuration* of a counter program with pushdown stack consists, as expected, of a valuation of its counters plus a stack content.

Counters are only allowed to have nonnegative values.

Conventions. We are particularly interested in counter programs *without zero tests*, i.e., ones that use no zero test command. In the sequel, unless specified explicitly, counter programs are implicitly assumed to be <u>without zero tests</u>. Moreover, we use the syntactic sugar **loop**, which iterates a sequence of command a nondeterministic number of times (see Example 6). Finally, we write consecutive increments and decrements of different variables on a single line and we use the following shorthands:

$\mathsf{x} \mathrel{+}= \mathsf{m}$	(increment the counter x by m)
$\mathsf{x} \mathrel{-}= \mathsf{m}$	(decrement the counter x by m)
$\mathsf{x} \longrightarrow \mathsf{y}$	(decrement the counter x by one and increment the counter y by one)
$\mathsf{x} \xrightarrow{\mathsf{m}} \mathsf{y}$	(decrement the counter x by m and increment the counter y by m)

▶ **Example 6.** As an illustration, consider three different presentations of the same the program with three counters $\mathsf{C} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$:

```
1: goto 2 or 6
2: x -= 1
3: y += 1
4: z += 2
5: goto 1 or 1
6: z += 1
```

```
1: loop
2:     x -= 1
3:     y += 1
4:     z += 2
5: z += 1
```

```
1: loop  x ⟶ y   z += 2
2: z += 1
```

The program repeats the block of commands in lines 2–4 some number of times chosen nondeterministically (possibly zero, although not infinite because x is decreasing, and hence its initial value bounds the number of iterations) and then increments z.

We emphasise that counters are only permitted to have nonnegative values. In the program above, that is why the decrement in line 2 works also as a non-zero test.

Runs. Consider a program with counters X. By $\mathbb{N}^{\mathsf{X}}$ we denote the set of all valuations of counters. Given an initial valuation of counters, a *run* (or *execution*) of a counter program is a finite sequence of executions of commands. A run which has successfully finished we call *complete*; otherwise, the run is *partial*. Observe that, due to a decrement that would cause a counter to become negative, a partial run may fail to continue because it is blocked from further execution. Moreover, due to nondeterminism of **goto**, a program may have various runs from the same initial valuation.

Two programs $\mathcal{P}, \mathcal{Q}$ may be *composed* by concatenating them, written $\mathcal{P}\ \mathcal{Q}$. We silently assume the appropriate re-numbering of lines referred to by **goto** command in $\mathcal{Q}$.

Consider a distinguished set of counters $\mathsf{Z} \subseteq \mathsf{X}$. A run of $\mathcal{P}$ is Z-*zeroing* if it is complete and all counters from Z are zero at the end. Given two counter valuations $r, r' \in \mathbb{N}^{\mathsf{X}}$ we say that the program Z-*computes* the valuation r from r' if $\mathcal{P}$ has exactly one Z-zeroing run from r' and the end configuration is r. We also say that the program Z-*computes nothing* from r' if $\mathcal{P}$ has no Z-zeroing run from r'. For instance, in Example 6 the programs $\{\mathsf{x}\}$-compute, from any valuation satisfying $\mathsf{x} = \mathsf{n} \in \mathbb{N}$ and $\mathsf{y} = \mathsf{z} = 0$, the valuation satisfying $\mathsf{x} = 0$ (trivially), $\mathsf{y} = \mathsf{n}$ and $\mathsf{z} = 2\mathsf{n} + 1$.

Maximal iteration. The proofs of this paper often focus on the number of iterations of the **loop** construct. Consider a program $\mathcal{P}$ containing a *flat* **loop**, i.e., a loop whose body consists of only increment or decrement commands, such that each counter appears in at most one of these commands (like the programs in Example 6). We say that this loop is *maximally iterated* in a given run of a $\mathcal{P}$ if some counter that is decremented in its body is zero at the exit from the loop. In particular, a maximally iterated loop could not be iterated any further without violation of the nonnegativity constraint on the decremented counter. For instance, the loop in Example 6 is maximally iterated by the $\{\mathsf{x}\}$-zeroing runs. Needless to say, maximal iteration needs not happen in general, for instance the program in Example 6 has multiple complete runs that do not admit this property.

3 Main results and structure of the paper

Counter programs (without zero test) provide an equivalent presentation to the standard models of Petri nets and VASS. The transformations between these different models are straightforward and the blowups are polynomial. For instance, a program can be transformed into a VASS by taking one state for each line of the program, and adding an appropriate transition corresponding to each counter update instruction. Note that the dimension of the VASS obtained is equal to the number of counters of the program. In this paper, we focus solely on counter programs, and we prove that the following problem is $\mathcal{F}_d$-hard for every $d \geq 3$:

▶ **Problem 1.**
Input: *A program $\mathcal{P}$ using $2d + 3$ counters.*
Question: *Is there a complete run of $\mathcal{P}$ that starts and ends with all counters equal to 0?*

The equivalence between programs and VASS then directly leads to our first main result, Theorem 1.

For programs with a pushdown stack, $\mathcal{F}_d$-hardness can be achieved with less counters. We show that the following problem is $\mathcal{F}_d$-hard for every $d \geq 3$:

▶ **Problem 2.**
Input: *A program with stack $\mathcal{Q}$ using $\lfloor \frac{d}{2} \rfloor + 6$ counters.*
Question: *Is there a complete run of $\mathcal{Q}$ that starts and ends with all counters equal to 0?*

Again, the equivalence between programs and VASS yields our second main result, Theorem 3.

Let us now introduce the known $\mathcal{F}_d$-hard problem that we will reduce to Problems 1 and 2. Fortunately, we do not have to search too far for it: counter programs with two counters *and zero tests* are Turing-complete [17]. This implies that the reachability problem is undecidable in that setting. However, similarly to Turing machines, decidability is recovered by imposing limitations on the executions, such as bounding their length, the maximal counter size, or the number of zero tests. For our purposes the latter is the easiest to use, thus, we present the problem that we will reduce from, a variation of the "$\mathbf{F}_k$-bounded Minsky Machine Halting Problem" proved to be $\mathcal{F}_d$-complete in [18, Section 2.3.2]:

▶ **Problem 3.**
Input: *A program $\mathcal{P}$ with two zero-tested counters, and a bound $n \in \mathbb{N}_+$.*
Question: *Is there a complete run of $\mathcal{P}$ that starts and ends with all counters equal to 0 and does exactly $\mathbf{F}_d(n)$ zero tests?*

The rest of this section is devoted to the presentation of the tools we use to reduce Problem 3 to Problems 1 and 2. Following the structure of similar reductions presented in [5] and [10], our reduction is divided into two main steps.

In the first step, we show how a program *without zero test* can simulate a bounded number of zero tests. To achieve this we rely on the concept of *triples*, which are specific counter valuations that allow to eliminate the zero tests and instead verify whether a particular invariant still holds at the term of the run. However, doing so requires an initial triple directly proportional to the number of zero tests we aim to simulate. Since we intend to simulate $\mathbf{F}_d(n)$ zero tests, which is a rather large number, directly applying this approach would result in an excessively large program.

Thus, in the second step, we construct compact *amplifiers*. These amplifiers are small programs that compute functions of substantial magnitude (such as $\mathbf{F}_d$) while using a small number of counters (namely $2d + 4$, or $\lfloor \frac{d}{2} \rfloor + 4$ counters along with a stack).

We now define formally the notions required for these two steps. This will allow us to state the main lemmas proved in this paper, and use them to construct the reduction proving our main result. The proofs of the lemmas can then be found in the following sections.

Triples. The concept of *triples* plays a central role in all the constructions presented within this paper. Given a set of counters X, three distinguished counters $\mathsf{a}, \mathsf{b}, \mathsf{c} \in \mathsf{X}$ and $A, B \in \mathbb{N}$, we denote by $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ the counter valuation satisfying

$$\mathsf{a} = \mathsf{A}, \quad \mathsf{b} = \mathsf{B}, \quad \mathsf{c} = \mathsf{A} \cdot (4^{\mathsf{B}} - 1), \quad \mathsf{x} = 0 \text{ for every } \mathsf{x} \in \mathsf{X} \setminus \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}. \tag{2}$$

Informally we sometimes call such valuation a *B-triple*, or simply a *triple* over the counters $\mathsf{a}, \mathsf{b}, \mathsf{c}$. The interest of triples lies in their ability to establish invariants that enable the detection of unwanted behaviours in counter programs. For instance, in Section 4, we show how to use triples to replace zero tests by proving the following lemma:

▶ **Lemma 7.** *Let $\mathcal{P}$ be a program of size m using two zero-tested counters. There exists a program $\mathcal{P}'$ of size $\mathcal{O}(m)$ with six counters X such that for every $B \in \mathbb{N}_+$ the two following conditions are equivalent:*

- *There exists a complete run of $\mathcal{P}$ that starts and ends with all counters equal to 0 and performs exactly B zero tests;*
- *There exists a complete run of $\mathcal{P}'$ that starts in some configuration $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ with $A \in \mathbb{N}_+$ and ends in a configuration where all the counters except a are zero.*

Amplifiers. The key contribution of this paper consists of the construction of two families of programs that transform B-triples into $\mathbf{F}_d(B)$-triples. We formalise this type of programs through the notion of *amplifiers*. Let $F : \mathbb{N}_+ \to \mathbb{N}_+$ be a monotone function satisfying $F(n) \geq n$ for every $n \in \mathbb{N}_+$. Consider a program $\mathcal{P}$ using the set of counters X, out of which we distinguish six counters $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{b}', \mathsf{c}', \mathsf{t} \in \mathsf{X}$, and a subset of counters $\mathsf{Z} \subseteq \mathsf{X} \setminus \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{b}', \mathsf{t}\}$ that contains c'. The tuple $(\mathcal{P}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}', \mathsf{c}'), \mathsf{t}, \mathsf{Z})$ is called *F-amplifier* if for all $A, B \in \mathbb{N}_+$:

- $\mathcal{P}$ Z-computes $\textsc{Triple}(A \cdot 4^{B-F(B)}, F(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ from $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ if A is divisible by $4^{(F(B)-B)}$;
- $\mathcal{P}$ Z-computes nothing from $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ if A is not divisible by $4^{(F(B)-B)}$.

An amplifier transforms B-triples on its *input counters* $\mathsf{a}, \mathsf{b}', \mathsf{c}'$ into $F(B)$-triples on its *output counters* $\mathsf{a}, \mathsf{b}, \mathsf{c}$. Remark that the counter a is involved in both input and output. The counters in Z, called *end counters*, are intuitively speaking assumed to be 0-checked after the completion of a run of $\mathcal{P}$. The auxiliary counter t does not play a direct role apart from *not* being an input counter, an output counter nor an end counter. This will prove useful in our constructions. We note that no condition is imposed on the runs that start from a counter valuation that is not a triple on the input counters, nor on the runs that are not Z-zeroing.

In Section 5 we construct of a family of $\mathbf{F}_d$-amplifiers:

▶ **Lemma 8.** *For every $d \in \mathbb{N}_+$ there exists an $\mathbf{F}_d$-amplifier of size $\mathcal{O}(d)$ that uses $2d + 4$ counters out of which d are end counters.*

Furthermore, in Section 6 we extend the notion of amplifiers to programs with stack, and show how a stack can replace three quarters of the counters used in the previous construction:

▶ **Lemma 9.** *For every $d \in \mathbb{N}_+$ there exists an $\mathbf{F}_d$-amplifier of size $\mathcal{O}(d)$ that uses a stack and $\lfloor \frac{d}{2} \rfloor + 4$ counters out of which $\lfloor \frac{d}{2} \rfloor$ are end counters.*

Proof of the main theorems. While the proofs of our key lemmas are delegated to the appropriate sections, we can already show how these lemmas yield a reduction from Problem 3 to Problems 1 and 2. Let us consider an instance of Problem 3, that is, a 2-counter program with zero tests $\mathcal{P}$ and an integer $n \in \mathbb{N}_+$. We transform this instance into an instance $\mathcal{P}''$ of Problem 1 and an instance $\mathcal{Q}''$ of Problem 2. These two programs rely on the program $\mathcal{P}'$ given by Lemma 7, and the $\mathbf{F}_d$-amplifiers $\mathcal{P}_d$ and $\mathcal{Q}_d$ given by Lemmas 8 and 9.[1]

Program $\mathcal{P}''$:

```
1: b' += n
2: loop a += 1    c' += 4^n − 1
3: P_d
4: P'
5: loop a −= 1
```

Program $\mathcal{Q}''$:

```
1: PUSH((b')^n)
2: loop a += 1    c' += 4^n − 1
3: Q_d
4: P'
5: loop a −= 1
```

[1] Note that, as explained in Section 6, when d is odd calling the program $\mathcal{Q}_d$ requires non-deterministically pushing the content of c' onto the stack along with the content of b', which we omit here.

Due to size estimations of Lemmas 7, 8 and 9, the sizes of programs $\mathcal{P}''$ and $\mathcal{Q}''$ are linear in the size of P plus $n + d$. We now prove that this is a valid reduction from Problem 3 to Problem 1. The proof for Problem 2 is analogous since the programs $\mathcal{P}_d$ and $\mathcal{Q}_d$ have identical effect on triples.

We need to show that $\mathcal{P}''$ has a complete run that starts and ends with all counters equal to 0 if and only if $\mathcal{P}$ has a complete run that starts and ends with all counters equal to 0 and that does exactly $\mathbf{F}_d(n)$ zero tests. In order to prove it, let us consider the structure of a hypothetical run π of $\mathcal{P}''$ that starts and ends with all counters having value zero. Starting from the configuration where all the counters are zero, Lines 1–2 generate an arbitrary n-triple: Progressing through line 1 and performing $A \in \mathbb{N}_+$ iterations of line 2 results in the configuration $\textsc{Triple}(A, n, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$. Next, it is important to note that the subrun of π involving $\mathcal{P}_d$ zeroes all the end counters, since these counters remain unchanged after the invocation of $\mathcal{P}_d$ and they have value zero at the end of π. Consequently, according to the definition of an amplifier, $\mathcal{P}_d$ transforms the n-triple $\textsc{Triple}(A, n, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ into an $\mathbf{F}_d(n)$-triple $\textsc{Triple}(A', \mathbf{F}_d(n), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$. From there, the subrun involving $\mathcal{P}'$ must end in a configuration where every counter except a has value zero since the final line of $\mathcal{P}''$ can only decrement a. Therefore, the run π exists if and only if there exists a run of $\mathcal{P}'$ that bridges the gap, starting from an $\mathbf{F}_d(n)$-triple $\textsc{Triple}(A', \mathbf{F}_d(n), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ and ending in a configuration where all the counters except a are 0. By Lemma 7 we know that such a run of $\mathcal{P}'$ exists if and only if $\mathcal{P}$ has a complete run that starts and ends with all counters equal to 0 and does exactly $\mathbf{F}_d(n)$ zero tests, which shows that our reduction is valid.

To conclude, let us remark that, as defined here, the program $\mathcal{P}''$ uses $2d + 7$ counters: the call to $\mathcal{P}_d$ requires $2d + 4$ counters, and while $\mathcal{P}'$ shares the output counters a, b and c of $\mathcal{P}_d$, it uses three more counters. We now argue that four counters can be saved, so that our program matches the definition of Problem 1.

- The value of the input counter b' is never incremented in the program $\mathcal{P}_d$ we construct.[2] Therefore, since in $\mathcal{P}''$ the call to $\mathcal{P}_d$ *always* starts with the value $\mathsf{b}' = n$, we can get rid of the counter b' by replacing the call to $\mathcal{P}_d$ with n consecutive copies of $\mathcal{P}_d$ in which each instruction decrementing b' is replaced with a jump to the next copy.
- The second optimisation consists in reusing the counters of $\mathcal{P}_d$. Since $\mathcal{P}_d$ is an amplifier, at the term of the run it is sufficient to check that the end counters are 0 to ensure that *all* the counters except the output counters a and c are 0. Therefore, while the call to $\mathcal{P}'$ in $\mathcal{P}''$ needs to keep the d end counters of $\mathcal{P}_d$ untouched, there are still d freely reusable counters (not counting a, b and c that are already reused), and we can pick any three of these to use in $\mathcal{P}'$ instead of adding fresh ones.

For the program $\mathcal{Q}''$ it is not possible to save counters is that way, but one of the extra counters of $\mathcal{P}'$ can be loaded on the stack, which results in a program with $\lfloor \frac{d}{2} \rfloor + 6$ counters.

4 Triples as a replacement for zero tests

The goal of this section is to prove Lemma 7.

The six counters of program $\mathcal{P}'$ will consists in two counters x, y simulating the two counters of $\mathcal{P}$, three counters $\mathsf{a}, \mathsf{b}, \mathsf{c}$ containing the initial triple, and an auxiliary counter t. The idea behind the construction is that we will replace the zero tests with the two gadgets $\mathcal{Z}ero(\mathsf{x})$ and $\mathcal{Z}ero(\mathsf{y})$ defined as follows:

[2] Remark that this is *not* stated explicitly by Lemma 8, but it is a trivial property of the corresponding construction presented in Section 5.

Program $\mathcal{Z}ero(\mathsf{x})$**:**

1: **loop** $\mathsf{a} \longrightarrow \mathsf{t}$ $\mathsf{c} \longrightarrow \mathsf{t}$
2: **loop** $\mathsf{y} \longrightarrow \mathsf{x}$ $\mathsf{c} \longrightarrow \mathsf{t}$
3: **loop** $\mathsf{t} \longrightarrow \mathsf{a}$ $\mathsf{c} \longrightarrow \mathsf{a}$
4: **loop** $\mathsf{x} \longrightarrow \mathsf{y}$ $\mathsf{c} \longrightarrow \mathsf{a}$
5: $\mathsf{b} -= 1$

Program $\mathcal{Z}ero(\mathsf{y})$**:**

1: **loop** $\mathsf{a} \longrightarrow \mathsf{t}$ $\mathsf{c} \longrightarrow \mathsf{t}$
2: **loop** $\mathsf{x} \longrightarrow \mathsf{y}$ $\mathsf{c} \longrightarrow \mathsf{t}$
3: **loop** $\mathsf{t} \longrightarrow \mathsf{a}$ $\mathsf{c} \longrightarrow \mathsf{a}$
4: **loop** $\mathsf{y} \longrightarrow \mathsf{x}$ $\mathsf{c} \longrightarrow \mathsf{a}$
5: $\mathsf{b} -= 1$

The functioning of these two programs revolves around the following invariant:

INVARIANT: $(\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}) \cdot 4^{\mathsf{b}} = \mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}+\mathsf{c}$ and $\mathsf{t} = 0$;

BROKEN INVARIANT: $(\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}) \cdot 4^{\mathsf{b}} < \mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}+\mathsf{c}$.

Remark that the broken invariant is more specific than the negation of the invariant. We now present a technical lemma showing that $\mathcal{Z}ero(\mathsf{x})$ and $\mathcal{Z}ero(\mathsf{y})$ accurately replace zero tests.

▶ **Lemma 10.** *Let* $\mathsf{z} \in \{\mathsf{x}, \mathsf{y}\}$*. From each configuration of* $\mathcal{Z}ero(\mathsf{z})$ *where* $\mathsf{b} > 0$*,* $\mathsf{z} = 0$ *and the invariant holds there is a unique complete run that maintains the invariant and preserves the values of* x *and* y*. All other runs starting from this configuration, as well as all runs starting with a broken invariant or a holding invariant with a value of* z *greater than* 0*, end with a broken invariant.*

Proof. We show the result for $\mathcal{Z}ero(\mathsf{x})$: the proof can easily be transferred to $\mathcal{Z}ero(\mathsf{y})$ by exchanging the roles of x and y. Let us start by observing that $\mathcal{Z}ero(\mathsf{x})$ globally decrements the value of b by 1 and preserves the sum $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}+\mathsf{c}$ since every line preserves it. Therefore, in order to maintain a holding invariant, $\mathcal{Z}ero(\mathsf{x})$ needs to quadruple the value of the sum of counters $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$. Similarly, to repair a broken invariant $\mathcal{Z}ero(\mathsf{x})$ needs to increase the value of $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$ by more than quadrupling it. We now study $\mathcal{Z}ero(\mathsf{x})$ in detail to show that the latter is impossible, and that the former only occurs under specific conditions that imply the statement of the lemma. We split our analysis in two:

Lines 1–2: The loop on line 1, respectively 2, increases $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$ by at most a, respectively y. Therefore the value of $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$ is at most doubled, which occurs only if initially $\mathsf{t} = \mathsf{x} = 0$ and if then both loops are maximally iterated, resulting in $\mathsf{a} = \mathsf{y} = 0$.

Lines 3–4: The loop on line 3, respectively 4, increases $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$ by at most t, respectively x. Therefore the value of $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$ is at most doubled, which happens only if $\mathsf{a} = \mathsf{y} = 0$ upon reaching line 3 and if then both loops are maximally iterated, resulting in $\mathsf{t} = \mathsf{x} = 0$.

Combining both parts, we get that the value of $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$ is at most quadrupled through a run of $\mathcal{Z}ero(\mathsf{x})$, which only occurs if initially $\mathsf{t} = \mathsf{x} = 0$ and all the loops are maximally iterated. This immediately implies that $\mathcal{Z}ero(\mathsf{x})$ cannot repair a broken invariant. Moreover, to maintain a holding invariant it is necessary to actually quadruple this value, therefore x needs to be 0 at the start; the maximal iteration of the loops will cause t to be 0 at the end; and the content of y will be completely moved to x by line 2 and then back to y by line 4, remaining unchanged as required by the statement. ◀

We now use $\mathcal{Z}ero(\mathsf{x})$ and $\mathcal{Z}ero(\mathsf{y})$ to transform $\mathcal{P}$ into a program $\mathcal{P}'$ satisfying Lemma 7. Formally, the program $\mathcal{P}'$ is obtained by applying the following modifications to $\mathcal{P}$:

- We add an increment (resp. decrement) of a to each line of $\mathcal{P}$ featuring a decrement (resp. increment) of x or y so that every line preserves the value of $\mathsf{a}+\mathsf{x}+\mathsf{y}+\mathsf{t}$;
- We replace each zero tests "**zero?** x" with a copy of the program $\mathcal{Z}ero(\mathsf{x})$, and each zero test "**zero?** y" with a copy of the program $\mathcal{Z}ero(\mathsf{y})$;

The first modification ensures that all the lines of $\mathcal{P}'$ except the calls to $\mathcal{Z}ero(\mathsf{x})$ and $\mathcal{Z}ero(\mathsf{y})$ maintain the invariant. We observe that for every complete run π' of $\mathcal{P}'$ that starts in a triple $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ and ends in a configuration where all counters except a are zero, the invariant is satisfied both at the beginning (by the definition of a triple) and at the end. Therefore, according to Lemma 10 the invariant is never broken throughout π', indicating that the calls to $\mathcal{Z}ero(\mathsf{x})$ and $\mathcal{Z}ero(\mathsf{y})$ accurately simulate zero tests. Consequently, π' can be transformed into a matching run π of $\mathcal{P}$ with same values of x and y. In particular, π starts and ends with both counters equal to 0. Additionally, the value of b goes from B to 0 along π'. Since this value is decremented by one by each call to $\mathcal{Z}ero(\mathsf{x})$ or $\mathcal{Z}ero(\mathsf{y})$, π' goes through exactly B such calls, which translates into π performing exactly B zero tests.

To conclude the proof of Lemma 7, remark that we can also transform every run of $\mathcal{P}$ that starts and ends with both counters equal to 0 and performs B zero tests into a matching run of $\mathcal{P}'$ starting from some triple $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ and ending in a configuration where all counters except a are zero However, we must be cautious in choosing the initial value A of a to be sufficiently high. This ensures that we can increment x and y as high as required, despite the matching decrements of a added in $\mathcal{P}'$.

The size of $\mathcal{P}'$ is linear in the size of $\mathcal{P}$, as required.

5 Amplifiers defined by counter programs

This section is devoted to an inductive proof of Lemma 8.

First we build an $\mathbf{F}_1$-amplifier $\mathcal{P}_1$ with 6 counters out of which one is an end counter (Lemma 11). Next we show how to lift an arbitrary F-amplifier with d counters into an $\widetilde{F}$-amplifier by adding two counters out of which one is an end counter (Lemma 14). Applying $d-1$ times our lifting process to the program $\mathcal{P}_1$ yields an $\mathbf{F}_d$-amplifier using $2d+4$ counters, proving Lemma 8.

Strong amplifiers. For the purpose of induction step, namely for lifting F-amplifiers to $\widetilde{F}$-amplifiers, we need a slight strengthening of the notion of amplifier. An F-amplifier $(\mathcal{P}, (\mathsf{a}, \mathsf{b}, \mathsf{b}), (\mathsf{a}, \mathsf{b}', \mathsf{c}'), \mathsf{t}, \mathsf{Z})$ is called *strong* if every run π of $\mathcal{P}$ satisfies the following conditions (let $\Sigma\mathsf{Z}$ stand for the sum of all counters in Z):

1. The value of the sum $\mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z}$ is the same at the start and at the end of π;
2. If $(\mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z}-\mathsf{c}')\cdot 4^{\mathsf{b}'} < \mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z}$ holds at the start of π, it also holds at the end.

5.1 Construction of the $\mathbf{F}_1$-amplifier $\mathcal{P}_1$

Consider the following program with 6 counters $\mathsf{X} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{b}', \mathsf{c}', \mathsf{t}\}$:

Program $\mathcal{P}_1$:

```
loop a ⟶ t
loop t ⟶ a    c' −= 3    c += 3
b' −= 1    b += 1
loop
    loop a ⟶ t    c' −= 1    t += 1
    loop c ⟶ a    c' −= 1    a += 1
    loop a ⟶ c    c' −= 1    c += 1
    loop t ⟶(8) c    c' −= 8    a += 1    c += 7
    b' −= 1    b += 2
```

The program consists of an *initialisation* step lines 1–3, and an *iteration* step lines 4–9. The initialisation step and the iteration step both decrement b' by one and preserve the right-hand side $\mathsf{a}+\mathsf{c}+\mathsf{t}+\mathsf{c}'$ of the invariant as every line of $\mathcal{P}_1$ preserves this sum. Hence, to maintain the invariant the sum $\mathsf{a}+\mathsf{c}+\mathsf{t}$ needs to be quadrupled, and to repair a broken invariant the sum $\mathsf{a}+\mathsf{c}+\mathsf{t}$ needs to be increased by an even larger amount. We show that in both steps the latter is impossible, and the former only happens if all loops are maximally iterated, which implies the modification of the counter a required by the statements.

▶ **Lemma 11.** *The program* $(\mathcal{P}_1, \mathsf{X}, (\mathsf{a},\mathsf{b},\mathsf{c}), (\mathsf{a},\mathsf{b}',\mathsf{c}'), \mathsf{t}, \{\mathsf{c}'\})$ *is a strong* $\mathbf{F}_1$*-amplifier.*

As an $\mathbf{F}_1$-amplifier, $\mathcal{P}_1$ is expected to map each input $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ such that $4^{\mathbf{F}_1(B)-B}$ divides A to the output $\textsc{Triple}(A \cdot 4^{B-\mathbf{F}_1(B)}, \mathbf{F}_1(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$. Since $\mathbf{F}_1(B) = 2B-1$, transforming the initial value $\mathsf{b}' = B$ into the final value $\mathsf{b} = \mathbf{F}_1(B)$ is easy: $\mathcal{P}_1$ first decrements b by 1 and increments b by 1 once (line 3), and then increments b by 2 whenever it decrements b' by 1 (line 9). It is more complicated to transform the initial value $\mathsf{a} = A$ into the final value $\mathsf{a} = A \cdot 4^{B-\mathbf{F}_1(B)}$: we need to divide $\mathbf{F}_1(B) - B = B-1$ times the content of a by 4. We prove that $\mathcal{P}_1$ does so by studying the following invariant:

$$\textsc{Invariant:} \qquad (\mathsf{a}+\mathsf{c}+\mathsf{t}) \cdot 4^{\mathsf{b}'} = \mathsf{a}+\mathsf{c}+\mathsf{t}+\mathsf{c}' \quad \text{and} \quad \mathsf{t} = 0;$$

$$\textsc{Broken invariant:} \qquad (\mathsf{a}+\mathsf{c}+\mathsf{t}) \cdot 4^{\mathsf{b}'} < \mathsf{a}+\mathsf{c}+\mathsf{t}+\mathsf{c}'.$$

Notice that saying that the invariant is broken is more specific than saying that the invariant does not hold. We now present two technical lemmas describing how the invariant evolves along both steps of $\mathcal{P}_1$.

▶ **Lemma 12.** *From each configuration where* $\mathsf{b}' > 0$, $\mathsf{c} = 0$ *and the invariant holds, there is a unique run through the initialisation step that maintains the invariant and preserves the value of* a*. All the other runs starting from this configuration, as well as the runs starting with a broken invariant, end with a broken invariant.*

Proof. The value of $\mathsf{a}+\mathsf{c}+\mathsf{t}$ is at most increased by $3 \cdot (\mathsf{a}+\mathsf{t})$ along the initialisation part, as line 1 preserves this sum and moves the content of a to t, then line 2 increases this sum by at most 3 times the value of t. Therefore $\mathsf{a}+\mathsf{c}+\mathsf{t}$ is at most quadrupled, which implies that the initialisation step cannot repair a broken invariant. Moreover, to maintain a holding invariant the program $\mathcal{P}_1$ needs to quadruple this sum. This happens if and only if initially $\mathsf{b}' > 0$, $\mathsf{c} = 0$ and $\mathsf{c}' \geq 3 \cdot (\mathsf{a}+\mathsf{t})$ (this last condition is implied by the invariant); and if then both loops are maximally iterated. Finally, remark that upon maximal iteration of the loops a and t keep their initial values, as required. ◀

▶ **Lemma 13.** *From each configuration where* $\mathsf{b}' > 0$, a *is divisible by* 4 *and the invariant holds, there is a unique run through the iteration step that maintains the invariant and divides the value of* a *by* 4*. All the other runs starting from this configuration, as well as those starting with a broken invariant or a holding invariant with a value of* a *not divisible by* 4*, end with a broken invariant.*

Proof. We divide our analysis of the iteration step in two parts:

Lines 5–6. The loop on line 5, respectively line 6, increases $\mathsf{a}+\mathsf{c}+\mathsf{t}$ by at most the value of a, respectively c. Therefore the value of $\mathsf{a}+\mathsf{c}+\mathsf{t}$ is at most doubled, which occurs only if initially $\mathsf{t} = 0$ and then both loops are maximally iterated, resulting in $\mathsf{c} = 0$.

Lines 7–8. The loop on line 7, respectively 8, increases $\mathsf{a}+\mathsf{c}+\mathsf{t}$ by at most the value of a, respectively t. Therefore the value of $\mathsf{a}+\mathsf{c}+\mathsf{t}$ is at most doubled, which occurs only if $\mathsf{c} = 0$ upon reaching line 7 and then both loops are maximally iterated, resulting in $\mathsf{t} = 0$.

Combining the two parts, we get that the value of $\mathsf{a}+\mathsf{c}+\mathsf{t}$ is at most quadrupled by the iteration step. This directly implies that it is not possible to repair a broken invariant. Moreover, to maintain a holding invariant this sum needs to be quadrupled. This happens if and only if at the start of the iteration step $\mathsf{b}' > 0$, a is divisible by 4, $\mathsf{t} = 0$ and $\mathsf{c}' \geq 3 \cdot (\mathsf{a}+\mathsf{c})$ (note that the last two conditions are implied by the invariant); and if then the four loops are maximally iterated.[3] To conclude, remark that maximally iterating lines 5 and 8 results in dividing the value of a by four: line 5 transfers twice the initial value of a to c, one eighth of which is then transferred back to a by line 8. ◀

We proceed with the proof of Lemma 11. Let $A, B \in \mathbb{N}$ and let π be a $\{c'\}$-zeroing run of the program $\mathcal{P}_1$ starting from $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$. Initially the counters satisfy:

$$\mathsf{a} = A, \quad \mathsf{b}' = B, \quad \mathsf{c}' = A \cdot (4^B - 1), \quad \mathsf{b} = \mathsf{c} = \mathsf{t} = 0.$$

This directly implies that the invariant holds at the beginning of π. Let us analyse the values of the counters at the end of π. We immediately get $\mathsf{c}' = 0$ since π is $\{c'\}$-zeroing. Note that this implies that the invariant cannot be broken as the counters always hold non-negative integer values. As a consequence, Lemmas 12 and 13 imply that the invariant still holds at the end of π, and that π is the unique $\{c'\}$-zeroing run of of $\mathcal{P}_1$ starting from $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$. To conclude the proof, we now show that at the end of π all the counters match $\textsc{Triple}(A \cdot 4^{B-\mathbf{F}_1(B)}, \mathbf{F}_1(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$:

$$\mathsf{a} = A \cdot 4^{B-\mathbf{F}_1(B)}, \quad \mathsf{b} = \mathbf{F}_1(B), \quad \mathsf{c} = A \cdot (4^B - 4^{B-\mathbf{F}_1(B)}), \quad \mathsf{b}' = \mathsf{c}' = \mathsf{t} = 0.$$

First, the invariant directly yields $\mathsf{t} = 0$, and also $\mathsf{b}' = 0$ by using the fact that $\mathsf{c}' = 0$. As b' starts with value B and is decremented by one along the initialisation step and each iteration step, we get that π visits the iteration step $B - 1$ times. In turn, this implies that the final value of b is $2B - 1 = \mathbf{F}_1(B)$, and by Lemmas 12 and 13 we also get that the final value of a is $\frac{A}{4^{B-1}} = A \cdot 4^{B-\mathbf{F}_1(B)}$. Combining this with the fact that the initial value $A \cdot 4^B$ of the sum $\mathsf{a}+\mathsf{c}+\mathsf{t}+\mathsf{c}'$ is preserved along π finally yields the appropriate value for c.

Note that the run π exists if and only if $4^{\mathbf{F}_1(B)-B}$ divides A. Otherwise Lemma 13 implies that the invariant is broken before the end of the run. This proves that $\mathcal{P}_1$ is an $\mathbf{F}_1$-amplifier. The fact that $\mathcal{P}_1$ is a *strong* $\mathbf{F}_1$-amplifier then directly follows from Lemmas 12 and 13.

5.2 Construction of the $\widetilde{F}$-amplifier $\widetilde{\mathcal{P}}$ from an F-amplifier $\mathcal{P}$

Let $(\mathcal{P}, \mathsf{X}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}', \mathsf{c}'), \mathsf{t}, \mathsf{Z})$ be a strong F-amplifier for some function $F : \mathbb{N}_+ \to \mathbb{N}_+$. We construct a strong $\widetilde{F}$-amplifier $\widetilde{\mathcal{P}}$ out of $\mathcal{P}$. The program $\widetilde{\mathcal{P}}$ uses the counters of $\mathcal{P}$ plus two fresh input counters b'' and c'', and it shares the output counters of $\mathcal{P}$:

Program $\widetilde{\mathcal{P}}$:

```
1:  loop a ⟶ t
2:  loop t ⟶ a    c'' −= 3    c += 3
3:  b'' −= 1    b += 1
4:  loop
5:      loop a ⟶ t
6:      loop t ⟶ a    c'' −= 3    a += 3
7:      loop c ⟶ c'    c'' −= 3    c' += 3
8:      loop b ⟶ b'
9:      P
10:     b'' −= 1
```

[3] The fact that a is divisible by four is what allows line 8 to be maximally iterated: if it is not the case, the run would eventually get stuck with a content of t smaller than eight but greater than zero.

Similarly to $\mathcal{P}_1$ the program $\widetilde{\mathcal{P}}$ consists of an *initialisation* step lines 1–3 (differing from $\mathcal{P}_1$ only by renaming counters), and an *iteration* step lines 4–10 (differing significantly from $\mathcal{P}_1$).

▶ **Lemma 14.** *For every strong F-amplifier $(\mathcal{P}, \mathsf{X}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}', \mathsf{c}'), \mathsf{t}, \mathsf{Z})$, the program $(\widetilde{\mathcal{P}}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}'', \mathsf{c}''), \mathsf{Z} \cup \{\mathsf{c}''\})$ is a strong $\widetilde{F}$-amplifier.*

The proof of Lemma 14 can be found in Appendix A. Here, we provide an overview of the main intuition behind it. As an $\widetilde{F}$-amplifier, $\widetilde{\mathcal{P}}$ is expected to map each input $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}'', \mathsf{c}'', \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$ such that $4^{\widetilde{F}(B)-F(B)}$ divides A to the output $\textsc{Triple}(A \cdot 4^{F(B)-\widetilde{F}(B)}, \widetilde{F}(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$. The intended behaviour of $\widetilde{\mathcal{P}}$ is straightforward: Since for all $n \in \mathbb{N}_+$ the value $\widetilde{F}(n)$ is obtained by applying the function F to 1 for $n-1$ consecutive times, we expect $\widetilde{\mathcal{P}}$ to apply the program $\mathcal{P}$ exactly $B-1$ times to transform a 1-triple into an $\widetilde{\mathcal{F}}(B)$-triple. To show that $\widetilde{\mathcal{P}}$ behaves as expected, we study the following invariant:

$$\textsc{Invariant:} \qquad (\mathsf{a}+\mathsf{c}+\mathsf{t}) \cdot 4^{\mathsf{b}''} = \mathsf{a}+\mathsf{c}+\mathsf{t}+\mathsf{c}'' \text{ and } \mathsf{t}=0;$$

$$\textsc{Broken invariant:} \qquad (\mathsf{a}+\mathsf{c}+\mathsf{t}) \cdot 4^{\mathsf{b}''} < \mathsf{a}+\mathsf{c}+\mathsf{t}+\mathsf{c}''.$$

The starting configuration $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}'', \mathsf{c}'', \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$ satisfies the invariant. The program $\widetilde{\mathcal{P}}$ is designed such that every run π starting from such a configuration then satisfies:

- If along π all the loops are maximally iterated and all the calls to $\mathcal{P}$ are Z-zeroing, then the invariant holds until the end of π. Moreover, π then matches the expected behaviour of $\widetilde{\mathcal{P}}$ described above. In particular, π will correctly amplify $B-1$ times via $\mathcal{P}$ a triple $\textsc{Triple}(A, 1, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$, thus ending in $\textsc{Triple}(A \cdot 4^{F(B)-\widetilde{F}(B)}, \widetilde{F}(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$.
- However, if π fails to maximally iterate one loop, or does a call to $\mathcal{P}$ that is not Z-zeroing, then the invariant is irremediably broken, which implies that π is not $(Z \cup \{c''\})$-zeroing.

This proves that $\widetilde{\mathcal{P}}$ is a strong $\widetilde{F}$-amplifier.

6 Amplifiers defined by counter programs with a pushdown stack

The goal of this section is to prove Lemma 9.

Our construction is based on the amplifiers from Section 5. The main idea is to "delegate" some counters to the stack. The stack alphabet consists exactly of those counters which are delegated, and the value of each delegated counter x corresponds to the number of occurrences of the symbol x on the stack. Therefore, "delegated counters" can be understood as a synonym of "stack symbols" in the sequel. This idea motivates the following definition. Let S and X be two disjoint sets of delegated, respectively non-delegated, counters. We define the function

$$h_{\mathsf{X},\mathsf{S}} : \mathbb{N}^{\mathsf{X}} \times \mathsf{S}^* \to \mathbb{N}^{\mathsf{X} \cup \mathsf{S}}$$

that maps a configuration, i.e., a valuation v of the non-delegated counters of X together with a stack content $s \in \mathsf{S}^*$, to a valuation of all the counters from $\mathsf{X} \cup \mathsf{S}$, as follows: $h_{\mathsf{X},\mathsf{S}}(v, s) = v'$, where $v'(\mathsf{x}) = v(\mathsf{x})$ for $\mathsf{x} \in \mathsf{X}$, and $v'(\mathsf{x})$ is the number of occurrences of x in s for $\mathsf{x} \in \mathsf{S}$.

Using this definition, we establish a notion of *simulation* between programs with or without stack. Given an F-amplifier $\mathcal{P}$ with set of counters $\mathsf{X} \cup \mathsf{S}$, we say that a counter program with a stack $\mathcal{Q}$ *simulates* $\mathcal{P}$ if it satisfies the two following conditions:

- For every $A, B \in \mathbb{N}_+$ such that A is divisible by $4^{(F(B)-B)}$ there exists a run of $\mathcal{Q}$ between two configurations x and y satisfying $h_{\mathsf{X},\mathsf{S}}(x) = \textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ and $h_{\mathsf{X},\mathsf{S}}(y) = \textsc{Triple}(A \cdot 4^{(B-F(B))}, F(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$.
- For every run of $\mathcal{Q}$ between two configurations x and y there exists a run of $\mathcal{P}$ between $h_{\mathsf{X},\mathsf{S}}(x)$ and $h_{\mathsf{X},\mathsf{S}}(y)$;

We say that such a program with stack $\mathcal{Q}$ is an F-amplifier.

The rest of this section is devoted to the proof of Lemma 9. We rely on the constructions of Section 5, and similarly proceed in two steps. First, we transform the $\mathbf{F}_1$-amplifier $\mathcal{P}_1$ into a program with stack $\mathcal{Q}_1$ that simulates $\mathcal{P}_1$ with four counters, as it delegates the two other counters to the stack (Lemma 15). Next, we adapt the constructions used to lift F-amplifiers into $\widetilde{F}$-amplifier. This time, we will have *two* constructions that can be applied alternatively: the first introduces one counter and one delegated counter, and the second introduces two delegated counters (Lemma 16). Therefore, for every $d \in \mathbb{N}$, starting with the program $\mathcal{Q}_1$ and applying alternatively our two lifting constructions yields a $\mathbf{F}_d$-amplifier with $\lfloor \frac{d}{2} \rfloor + 4$ counters (as the other counters are delegated to the stack), which proves Lemma 9.

6.1 Construction of the $\mathbf{F}_1$-amplifier $\mathcal{Q}_1$

Consider the following program with 4 counters $\mathsf{X} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{t}\}$ and the stack alphabet $\mathsf{S} = \{\mathsf{b}', \mathsf{c}'\}$, which is obtained from the program $\mathcal{P}_1$ defined in Section 5 by replacing each decrement on b' and c' by the corresponding pop operation:

Program $\mathcal{Q}_1$:

1: **loop** $\mathsf{a} \longrightarrow \mathsf{t}$
2: **loop** $\mathsf{t} \longrightarrow \mathsf{a}$ $\quad \textsc{pop}(\mathsf{c}'\mathsf{c}'\mathsf{c}')$ $\quad \mathsf{c} \mathrel{+}= 3$
3: $\textsc{pop}(\mathsf{b}')$ $\quad \mathsf{b} \mathrel{+}= 1$
4: **loop**
5: $\quad$ **loop** $\mathsf{a} \longrightarrow \mathsf{t}$ $\quad \textsc{pop}(\mathsf{c}')$ $\quad \mathsf{t} \mathrel{+}= 1$
6: $\quad$ **loop** $\mathsf{c} \longrightarrow \mathsf{a}$ $\quad \textsc{pop}(\mathsf{c}')$ $\quad \mathsf{a} \mathrel{+}= 1$
7: $\quad$ **loop** $\mathsf{a} \longrightarrow \mathsf{c}$ $\quad \textsc{pop}(\mathsf{c}')$ $\quad \mathsf{c} \mathrel{+}= 1$
8: $\quad$ **loop** $\mathsf{t} \xrightarrow{8} \mathsf{c}$ $\quad \textsc{pop}(\mathsf{c}'\mathsf{c}'\mathsf{c}'\mathsf{c}'\mathsf{c}'\mathsf{c}'\mathsf{c}')$ $\quad \mathsf{a} \mathrel{+}= 1$ $\quad \mathsf{c} \mathrel{+}= 7$
9: $\quad$ $\textsc{pop}(\mathsf{b}')$ $\quad \mathsf{b} \mathrel{+}= 2$

▶ **Lemma 15.** *The program $\mathcal{Q}_1$ simulates the $\mathbf{F}_1$-amplifier $(\mathcal{P}_1, \mathsf{X}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}', \mathsf{c}'), \mathsf{t}, \{\mathsf{c}''\})$.*

Proof. Let h denote the function $h_{\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{t}\},\{\mathsf{b}',\mathsf{c}'\}}$ that transforms configurations of $\mathcal{Q}_1$ into configurations of $\mathcal{P}_1$. The program $\mathcal{Q}_1$ is a constrained version of $\mathcal{P}_1$: every line is identical with the added restriction that lines 2, 3, 6 and 9 can only be fired if the appropriate symbol is at the top of the stack. Therefore, we immediately get the second condition required for $\mathcal{Q}_1$ to simulate $\mathcal{P}_1$: for every run π of $\mathcal{Q}_1$ between two configurations x and y, the run π' of $\mathcal{P}$ that starts in $h(x)$ and uses the same lines as π is a valid run of $\mathcal{P}$ that ends in $h(y)$.

To conclude, we show that we can transform the $\{\mathsf{b}', \mathsf{c}'\}$-zeroing runs of $\mathcal{P}_1$ (thus in particular the runs that witness the $\mathbf{F}_1$-amplifier behaviour) into runs of $\mathcal{Q}_1$. To do so we rely on the fact that the counters b' and c' are only decreasing along the runs of $\mathcal{P}_1$. Formally, given a $\{\mathsf{b}', \mathsf{c}'\}$-zeroing run π of $\mathcal{P}_1$ between two configurations x and y, let $u_\pi \in \{\mathsf{b}', \mathsf{c}'\}^*$ be the word listing, in order, the occurrences of the decrements of b' and c' along π. We define a configuration x' of $\mathcal{Q}_1$ as follows: the counters $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{t}$ match the content they have in the starting configuration x of π, and the stack content is the reverse of the word u_π so that the first letter of u_π is at the top of the stack. This definition guarantees that:

- We have $h(x') = x$. Indeed, since π is $\{\mathsf{b}', \mathsf{c}'\}$-zeroing, the value of the counters b' and c' in the initial configuration x is equal to the number of times these counters are decremented;
- There exists a run π' of $\mathcal{Q}_1$ that starts from x' and follows the same lines as π: whenever a popping instruction appears the adequate symbol will be at the top of the stack. As the lines of $\mathcal{P}_1$ and $\mathcal{Q}_1$ are analogous, the configuration y' reached by π' satisfies $h(y') = y$. ◀

6.2 Construction of the $\widetilde{F}$-amplifiers $\widetilde{\mathcal{Q}}$ and $\overline{\mathcal{Q}}$ from an F-amplifier $\mathcal{Q}$

In Section 5, we showed how to lift an F-amplifier $\mathcal{P}$ into an $\widetilde{F}$-amplifier $\widetilde{\mathcal{P}}$. We now show two different manners of adapting the construction of $\widetilde{\mathcal{P}}$ in order to lift a program with stack $\mathcal{Q}$ simulating $\mathcal{P}$ into a program with stack simulating $\widetilde{\mathcal{P}}$.

▶ **Lemma 16.** *Let $\mathcal{Q}$ be a program simulating a strong F-amplifier $(\mathcal{P}, \mathsf{X}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}', \mathsf{c}'), \mathsf{t}, \mathsf{Z})$ without delegating the counters $\mathsf{a}, \mathsf{b}, \mathsf{c}$ and t.*

- *If $\mathcal{Q}$ delegates b' but not c', then $\widetilde{\mathcal{Q}}$ simulates $(\widetilde{\mathcal{P}}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}'', \mathsf{c}''), \mathsf{Z} \cup \{\mathsf{c}''\})$ while delegating two input counters b'' and c'' in addition to the counters delegated by $\mathcal{Q}$.*
- *If $\mathcal{Q}$ delegates b' and c', then $\overline{\mathcal{Q}}$ simulates $(\widetilde{\mathcal{P}}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}'', \mathsf{c}''), \mathsf{Z} \cup \{\mathsf{c}''\})$ while delegating only one input counter b'' in addition to the counters delegated by $\mathcal{Q}$.*

Program $\widetilde{\mathcal{Q}}$:

```
loop a ⟶ t
loop t ⟶ a   POP(c''c''c'')   c += 3
POP(b'')   b += 1
loop
    loop a ⟶ t
    loop t ⟶ a  POP(c''c''c'')  a += 3
    loop c ⟶ c'  POP(c''c''c'')  c' +=
  3
    loop b −= 1   PUSH(b')
    Q
    POP(b'')
```

Program $\overline{\mathcal{Q}}$:

```
loop a ⟶ t
loop t ⟶ a   c'' −= 3   c += 3
POP(b'')   b += 1
loop
    loop a ⟶ t
    loop t ⟶ a   c'' −= 3   a += 3
    loop
        loop b −= 1   PUSH(b')
        c −= 1   c'' −= 3   PUSH(c')
        loop b −= 1   PUSH(b')
        PUSH(c')
        loop b −= 1   PUSH(b')
        PUSH(c')
        loop b −= 1   PUSH(b')
        PUSH(c')
        loop b −= 1   PUSH(b')
    Q
    POP(b'')
```

The proof of Lemma 16 can be found in Appendix B. To convey the intuition behind it we analyse the differences between the two programs. The main difference concerns the counters delegated to the stack: If $\mathcal{Q}$ delegates only b', then the starting configurations for the calls to $\mathcal{Q}$ are easy to setup as the stack simply contains a sequence of b'. Therefore $\mathcal{Q}$ can be lifted via $\widetilde{\mathcal{Q}}$ which delegates both b'' and c''. However, if $\mathcal{Q}$ delegates both b' and c', then the starting configurations required for the calls to $\mathcal{Q}$ are more complex: the stack needs to contain the symbols b' and c' in a specific order. This prevents us from delegating both b'' and c'' to the stack, thus we need to lift $\mathcal{Q}$ via $\overline{\mathcal{Q}}$ which delegates only b'' and keeps c'' as a standard counter. A second difference between $\widetilde{\mathcal{Q}}$ and $\overline{\mathcal{Q}}$ concerns the loops updating b' and c' in the iteration step. To understand what is happening here, let us have a look at what happens when we replace the push and pop instructions by increments and decrements:

```
1: loop c ⟶ c′   c″ −= 3   c′ += 3
2: loop b ⟶ b′
```

```
1: loop
2:     loop b ⟶ b′
3:     c −= 1   c′ += 1   c″ −= 3
4:     loop b ⟶ b′
5:     c′ += 1
6:     loop b ⟶ b′
7:     c′ += 1
8:     loop b ⟶ b′
9:     c′ += 1
10:    loop b ⟶ b′
```

While these two sequences of instructions are different, we can remark that their global effect is identical, in the sense that every counter update realisable by the left one is also realisable by the right one, and reciprocally. However, if b' and c' are delegated to the stack then the sequence of instruction on the right is more powerful, as it performs the same number of increments of b' and c', but *in any order*, which allows to create many different stack contents. This is required so that $\overline{\mathcal{Q}}$ can construct the stack contents needed to call $\mathcal{Q}$.

References

1. Mohamed Faouzi Atig and Pierre Ganty. Approximating Petri Net Reachability Along Context-free Traces. In *Proceedings of FSTTCS 2011*, volume 13 of *LIPIcs*, pages 152–163, 2011.
2. Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in Two-Dimensional Vector Addition Systems with States Is PSPACE-Complete. In *Proceedings of LICS 2015*, pages 32–43, 2015.
3. Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. In *Proceedings of STOC 2019*, pages 24–33. ACM, 2019.
4. Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The Reachability Problem for Petri Nets Is Not Elementary. *Journal of the ACM*, 68(1):7:1–7:28, 2021.
5. Wojciech Czerwinski and Lukasz Orlikowski. Reachability in Vector Addition Systems is Ackermann-complete. In *Proceedings of FOCS 2021*, pages 1229–1240. IEEE, 2021.
6. Wojciech Czerwinski and Lukasz Orlikowski. Lower Bounds for the Reachability Problem in Fixed Dimensional VASSes. In *Proceedings of LICS 2022*, pages 40:1–40:12. ACM, 2022.
7. Matthias Englert, Piotr Hofman, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Juliusz Straszynski. A lower bound for the coverability problem in acyclic pushdown VAS. *Inf. Process. Lett.*, 167:106079, 2021.
8. Matthias Englert, Ranko Lazić, and Patrick Totzke. Reachability in Two-Dimensional Unary Vector Addition Systems with States is NL-Complete. In *Proceedings of LICS 2016*, pages 477–484. ACM, 2016.
9. Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In *Proceeding of CONCUR 2009*, volume 5710, pages 369–383. Springer, 2009.
10. Slawomir Lasota. Improved Ackermannian Lower Bound for the Petri Nets Reachability Problem. In *Proceedings of STACS 2022*, volume 219 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
11. Jérôme Leroux. The reachability problem for petri nets is not primitive recursive. *CoRR*, abs/2104.12695, 2021. `arXiv:2104.12695`.

12 Jérôme Leroux. The Reachability Problem for Petri Nets is Not Primitive Recursive. In *Proceedings of FOCS 2021*, pages 1241–1252. IEEE, 2021.

13 Jérôme Leroux, M. Praveen, and Grégoire Sutre. Hyper-Ackermannian bounds for pushdown vector addition systems. In *Proceedings of CSL-LICS 2014*, pages 63:1–63:10. ACM, 2014.

14 Jérôme Leroux and Sylvain Schmitz. Reachability in Vector Addition Systems is Primitive-Recursive in Fixed Dimension. In *Proceedings of LICS 2019*, pages 1–13. IEEE, 2019.

15 Jérôme Leroux, Grégoire Sutre, and Patrick Totzke. On the Coverability Problem for Pushdown Vector Addition Systems in One Dimension. In *Proceedings of ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 324–336. Springer, 2015.

16 Richard J. Lipton. The Reachability Problem Requires Exponential Space. Technical report, Yale University, 1976.

17 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, 1967.

18 Sylvain Schmitz. Complexity Hierarchies beyond Elementary. *TOCT*, 8(1):3:1–3:36, 2016.

A Proof of Lemma 8

▶ **Lemma 14.** *For every strong F-amplifier $(\mathcal{P}, \mathsf{X}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}', \mathsf{c}'), \mathsf{t}, \mathsf{Z})$, the program $(\widetilde{\mathcal{P}}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\}, (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{b}'', \mathsf{c}''), \mathsf{Z} \cup \{\mathsf{c}''\})$ is a strong $\widetilde{F}$-amplifier.*

The proof is structured as follows: We begin with a technical lemma implying that $\widetilde{\mathcal{P}}$ satisfies the two invariants required to be a *strong* amplifier (Claim 17). Then, to show that $\widetilde{\mathcal{P}}$ is an $\widetilde{\mathcal{F}}$-amplifier, we formalise the expected behaviour of the runs of $\widetilde{\mathcal{P}}$ (Equations (3)–(7)), we show that the runs that fit this expected behaviour $(Z \cup \{c''\})$-compute $\widetilde{\mathcal{F}}$ (Claim 18), and that the runs that do not fit this expected behaviour are not $(Z \cup \{c''\})$-zeroing (Claim 19).

Invariants of $\widetilde{\mathcal{P}}$. Before delving into the intricate functioning of $\widetilde{\mathcal{P}}$, we show two invariants that hold for every run. On top of being prerequisites for $\widetilde{\mathcal{P}}$ to qualify as a *strong* amplifier, these invariants offer valuable assistance in proving the next results of this section.

▷ Claim 17. The initialisation step and the iteration step of $\widetilde{\mathcal{P}}$ both preserve the value of $\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z} + \mathsf{c}''$ and either preserve or decrease the value of $(\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}) \cdot 4^{\mathsf{b}''}$.

Proof. We start by observing that We start by observing that the sum $\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z} + \mathsf{c}''$ stays constant along every run of $\widetilde{\mathcal{P}}$: every command line preserves it, including line 9 since $\mathcal{P}$ is a strong F-amplifier. We now study the effect of the initialisation and iteration steps on the value of $(\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}) \cdot 4^{\mathsf{b}''}$.

Initialisation: The sum $\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}$ is preserved in lines 1 and 3 and is increased in line 2 by at most three times the value of t, thus it is at most quadrupled by the initialisation step. Since b'' is decremented by 1 during the initialisation step, this proves that the value of $(\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}) \cdot 4^{\mathsf{b}''}$ is either preserved or decreased.

Iteration: The sum $\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}$ is increased in lines 5–6 by at most three times the value of $\mathsf{a} + \mathsf{t}$; it is increased in line 7 by at most three times the value of c; and it is then preserved in lines 8, 9 and 10 (since $\mathcal{P}$ is a strong amplifier). Hence the value of $\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}$ is at most quadrupled by each occurrence of the iteration step. Since b'' is decremented by 1, this shows that the value of $(\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}) \cdot 4^{\mathsf{b}''}$ is either preserved or decreased. ◁

Expected behaviour of $\widetilde{\mathcal{P}}$. The intended behaviour of $\widetilde{\mathcal{P}}$ is straightforward. We start with a B-triple over the counters $\mathsf{a}, \mathsf{b}'', \mathsf{c}''$. The initialisation step establishes a 1-triple over $\mathsf{a}, \mathsf{b}, \mathsf{c}$. Next, in the iteration step, this 1-triple is first moved to $\mathsf{a}, \mathsf{b}', \mathsf{c}'$, and then $\mathcal{P}$ is invoked to transform it into a $F(1)$-triple over $\mathsf{a}, \mathsf{b}, \mathsf{c}$. By repeating the iteration step $B-2$ more times, we obtain a $F^{B-1}(1) = \widetilde{\mathcal{F}}(B)$-triple over $\mathsf{a}, \mathsf{b}, \mathsf{c}$, as expected from a $\widetilde{\mathcal{F}}$-amplifier. We now formalise this expected behaviour as a set of equations.

Let π be a run of $\widetilde{\mathcal{P}}$ that visits the iteration step n times. Let $w_0(\pi)$ denote the valuation of the counter set $\mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\}$ at the start of π, and $x_n(\pi)$ denote the valuation of the counter set X at the end of π. Moreover, for every $i = 0, 1 \ldots, n-1$, we use $x_i(\pi)$ and $y_i(\pi)$ to denote the valuation of X at the start of the $(i+1)$th iteration step of π and at the start of the $(i+1)$th call to the program $\mathcal{P}$, respectively. This notation allows us to formally express the expected behaviour described earlier:

$$w_0(\pi) = \textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}'', \mathsf{c}'', \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\}) \tag{3}$$

$$x_0(\pi) = \textsc{Triple}(A, 1, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X}) \tag{4}$$

$$x_i(\pi) = \textsc{Triple}(A \cdot 4^{i+1-F^i(1)}, F^i(1), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X}) \tag{5}$$

$$y_i(\pi) = \textsc{Triple}(A \cdot 4^{i+2-F^i(1)}, F^i(1), \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X}) \tag{6}$$

$$x_{B-1}(\pi) = \textsc{Triple}(A \cdot 4^{B-\widetilde{F}(B)}, \widetilde{F}(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X}) \tag{7}$$

The individual counter values corresponding to these equations are listed in Figure 1.

We split the set of runs of $\widetilde{\mathcal{P}}$ in two parts: the *good* runs, for which we show that Equations (3)–(7) hold, and the *bad* runs, that we prove to be non $(Z \cup \{c''\})$-zeroing. Formally, we say that a run of $\widetilde{\mathcal{P}}$ is *good* if it goes through $B-1$ iteration steps; if all the loops visited along it are maximally iterated; and if all its calls to the program $\mathcal{P}$ are Z-zeroing. By opposition, we describe as *bad* the runs that fail to satisfy at least one of these conditions.

Good runs. We prove that the good runs of $\widetilde{\mathcal{P}}$ compute the function $\widetilde{F}$:

▷ Claim 18. Let $A, B \in \mathbb{N}_+$ be two positive integers. Every good run of $\widetilde{\mathcal{P}}$ starting in $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}'', \mathsf{c}'', \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$ satisfies Equations (3)–(7), thus in particular it ends in $\textsc{Triple}(A \cdot 4^{B-\widetilde{F}(B)}, \widetilde{F}(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$. Moreover, there exists such a run if and only if $4^{\widetilde{F}(B)-B}$ divides A.

Proof. Let π be a good run of $\widetilde{\mathcal{P}}$ starting in $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}'', \mathsf{c}'', \mathsf{X})$. We immediately get that Equation (3) is satisfied. To show that π satisfies Equations (4)–(7), we prove via three inductive steps that Figure 1 is an accurate depictions of the valuations $x_i(\pi)$ and $y_i(\pi)$ for every $i \in \{0, 2, \ldots, B-1\}$:

	a	b	b'	c	c'
$w_0(\pi)$:	A	0	0	0	0
$x_0(\pi)$:	A	1	0	$A \cdot 2 - \mathsf{a}$	0
$x_i(\pi)$:	$A \cdot 4^{i+1-F^i(1)}$	$F^i(1)$	0	$A \cdot 4^{i+1} - \mathsf{a}$	0
$y_i(\pi)$:	$A \cdot 4^{i+2-F^i(1)}$	0	$F^i(1)$	0	$A \cdot 4^{i+2} - \mathsf{a}$
$x_{B-1}(\pi)$:	$A \cdot 4^{B-\widetilde{F}(B)}$	$\widetilde{F}(B)$	0	$A \cdot 4^B - \mathsf{a}$	0

Figure 1 Individual counter values corresponding to the expressions in Equations (3)–(7).

1. First, starting from a valuation satisfying $\mathsf{c}'' \geq \mathsf{a}$, the effect of the initialisation step with maximal iteration of the flat loops is equivalent to the following sequence of assignments:

$$\mathsf{b}'' := \mathsf{b}'' - 1, \qquad \mathsf{b} := \mathsf{b} + 1, \qquad \mathsf{c}'' := \mathsf{c}'' - \mathsf{a}, \qquad \mathsf{c} := \mathsf{a}.$$

   This maps the first row of Figure 1 to its second row, thus Equation (4) holds.
2. Next, starting from a valuation satisfying $\mathsf{c}'' \geq \mathsf{a}$, the effect of lines 5–8 of $\widetilde{\mathcal{P}}$ with maximal iteration of the flat loops is equivalent to the following sequence of assignments:

$$\mathsf{a} := 2 \cdot \mathsf{a}, \qquad \mathsf{b}' := \mathsf{b}, \qquad \mathsf{b} := 0, \qquad \mathsf{c}'' := \mathsf{c}'' - (\mathsf{a} + \mathsf{c}), \qquad \mathsf{c}' := \mathsf{c}' + 2\mathsf{c}, \qquad \mathsf{c} := 0.$$

   This maps the third row of Figure 1 to its fourth row, thus whenever Equation (5) holds for some $0 \in \{1, 2, \ldots, B-2\}$, so does Equation (6).
3. Finally, as the program $\mathcal{P}$ is a strong F-amplifier, for all $i \in \{0, 1, \ldots, B-2\}$ it Z-computes $\textsc{Triple}(A \cdot 4^{i+2-F^{i+1}(1)}, F^{i+1}(1), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ from $\textsc{Triple}(A \cdot 4^{i+2-F^{i}(1)}, F^{i}(1), \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$. Therefore, as every call to $\mathcal{P}$ along π is Z-zeroing since π a good run, we get that if Equation (6) holds for some $i \in \{0, 1, \ldots, B-2\}$ then Equation (5) holds for $i+1$.

To show that the run π ends in $\textsc{Triple}(A \cdot 4^{B-\widetilde{F}(B)}, \widetilde{F}(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$, we still need to address the values of counters b'' and c'' (as Equation 7 only specifies the value of the counter set X). We directly get that the value of b'' is 0 at the end of π: b'' starts with value B and is decremented once in the initialisation step and in each of the $B-1$ occurrences of the iteration step. Moreover, we also get that c'' is 0 at the end of π since Claim 17 yields that the value of $\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z} + \mathsf{c}''$ is constantly equal to $A \cdot 4^B$ along π.

Finally, concerning the existence of the run π, remark that, while the register updates mentioned in Item 1 and 2 can be applied irrespective of the values of A and B, the Z-zeroing calls to $\mathcal{P}$ described in Item 3 can be fulfilled if and only if A is divisible by a sufficiently large power of 2. More specifically, the run π described in this proof exists if and only if $4^{\widetilde{F}(B)-B}$ divides A. ◁

Bad runs. We prove that the bad runs of $\widetilde{\mathcal{P}}$ do not $(Z \cup \{c''\})$-compute anything:

▷ Claim 19. Let $A, B \in \mathbb{N}_+$ be two positive integers. Every bad run of $\widetilde{\mathcal{P}}$ starting in $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}'', \mathsf{c}'', \mathsf{X} \cup \{\mathsf{b}'', \mathsf{c}''\})$ is not $(Z \cup \{c''\})$-zeroing.

Proof. Let π be a run of $\widetilde{\mathcal{P}}$ starting in $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}'', \mathsf{c}'', \mathsf{X})$. At the start of π we have:

$$\mathsf{a} = \mathsf{A}, \quad \mathsf{b}'' = \mathsf{B}, \quad \mathsf{c}'' = \mathsf{a} \cdot (4^{\mathsf{b}''} - 1),$$

and all other counters are 0. In particular, this implies $\mathsf{b} = \mathsf{b}' = \mathsf{c} = \mathsf{c}' = \mathsf{t} = 0$, thus

$$(\mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z}) \cdot 4^{\mathsf{b}''} = \mathsf{a} + \mathsf{c} + \mathsf{t} + \Sigma\mathsf{Z} + \mathsf{c}''. \tag{8}$$

We start by observing that Claim 17 implies that π is Z-zeroing if and only if Equation (8) holds after every step of π and $b' = 0$ at the end of π: The right-hand of Equation (8) side is preserved, and the value of the left hand-side never increases, thus if it ever decreases it remains smaller than the right-hand side until the term of π, which in particular implies that the value of c'' is not 0.

Therefore we can immediately deduce that if π visits the iteration step less than $B-1$ times, then $b' > 0$ at the end of π, thus π is not $(Z \cup \{c''\})$-zeroing by Equation (8). For the rest of this proof, let us suppose that π visits the iteration step $B-1$ times. Whenever π goes through the initialisation step or the iteration step, it decrements b' by 1, while gaining

the opportunity to increment the sum $\mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z}$, that we denote $\Sigma\mathsf{Z}'$ in order to maintain Equation (8). As we showed when we studied the good runs, in an ideal scenario $\Sigma\mathsf{Z}'$ is quadrupled, which compensates the decrement of b', and Equation (8) still holds. We now show that the occurrence of a single mistake at any point results in $\Sigma\mathsf{Z}'$ not being quadrupled along a step: We suppose that the run π is bad, we list all the possible errors it can commit, and show that each one breaks Equation (8):

- If π fails to maximally iterate one of the flat loops at lines 1 or 2 then the sum $\Sigma\mathsf{Z}'$ is not quadrupled during the initialisation step: Maximally iterating both loops (that is, iterating both of them a times) increments $\Sigma\mathsf{Z}'$ by $3\cdot\mathsf{a}$, which exactly quadruples it since initially the other variables occurring in S have value 0. However, since line 2 increases $\Sigma\mathsf{Z}'$, not maximally iterating it results in a smaller value. Moreover, while line 1 has no direct effect on $\Sigma\mathsf{Z}'$, not maximally iterating it reduces the number of times line 2 can be iterated, which in turn reduces the value of $\Sigma\mathsf{Z}'$.
- If π fails to maximally iterate one of the flat loops at lines 5, 6 or 7 then the sum $\Sigma\mathsf{Z}'$ is not quadrupled during the corresponding iteration step: Maximally iterating the three loops (that is, iterating lines 5 and 6 a times and line 7 c times) increments $\Sigma\mathsf{Z}'$ by $3\cdot(\mathsf{a}+\mathsf{c})$, which exactly quadruples it as long as the other variables occurring in $\Sigma\mathsf{Z}'$ had value 0 to start with. However, since lines 6 and 7 increase $\Sigma\mathsf{Z}'$, not maximally iterating them results in a smaller value. Moreover, while line 5 has no direct effect on $\Sigma\mathsf{Z}'$, not maximally iterating it reduces the number of times line 6 can be iterated.
- If π does a non Z-zeroing call to the program $\mathcal{P}$, we differentiate two cases. If this happens in the last iteration step we get that π is not $(Z\cup\{c''\})$-zeroing as it is not Z-zeroing. If this happens in one of the previous iteration steps then the sum $\Sigma\mathsf{Z}'$ is not quadrupled in the *next* iterations step: as we just saw the iteration step increases S by at most $3\cdot(\mathsf{a}+\mathsf{c})$, which fails to quadruple it if there are nonzero counters in Z.
- Finally, let us consider the case where the first error committed by π is failing to maximally iterate the flat loop at line 8. In this case, we show that the subsequent call to $\mathcal{P}$ is not Z-zeroing, which, as we have just shown, implies that π is not $(Z\cup c'')$-zeroing. Since we assume that this is the first error committed by π, up to this point, π has behaved as a good run. To analyse this situation, let π' be the run that behaves as π up to this point but then maximally iterates the flat loop at line 8. By Lemma 18, we know that π' enters the call to $\mathcal{P}$ with a counter valuation matching $\textsc{Triple}(A\cdot 4^{i+2-F^i(1)}, F^i(1), \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ for some $0\le i\le B-1$. In particular, the following equation holds for π':

  $$(\mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z})\cdot 4^{\mathsf{b}'} = A\cdot 4^{i+2} = \mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z}+\mathsf{c}'$$

  However, since π did *not* maximally iterate line 8, b' will be smaller in π compared to π' (and b will be larger - but this has no impact on the following argument since $\mathsf{b}\notin\mathsf{Z}$). Consequently, π will call the program $\mathcal{P}$ with a counter valuation satisfying:

  $$(\mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z})\cdot 4^{\mathsf{b}'} < \mathsf{a}+\mathsf{c}+\mathsf{t}+\Sigma\mathsf{Z}+\mathsf{c}'.$$

  Since $\mathcal{P}$ is a strong amplifier, this equation still holds at the exit of $\mathcal{P}$. In particular, this implies that c' is not 0, thus the call to $\mathcal{P}$ is not Z-zeroing. ◁

B Proof of Lemma 16

Let X and S denote the set of counters of $\mathcal{Q}$, respectively its stack alphabet. Let $\widetilde{h}$ denote the function $h_{\mathsf{X},\mathsf{S}\cup\{\mathsf{b}'',\mathsf{c}''\}}$ that transforms configurations of $\widetilde{\mathcal{Q}}$ into configurations of $\widetilde{\mathcal{P}}$. Similarly, let $\overline{h}$ denote the function $h_{\mathsf{X}\cup\{\mathsf{c}''\},\mathsf{S}\cup\{\mathsf{b}''\}}$ that transforms configurations of $\overline{\mathcal{Q}}$

into configurations of $\widetilde{\mathcal{P}}$. The proof is done in three steps. First, we show that we can easily translate the runs of $\widetilde{\mathcal{Q}}$ and $\overline{\mathcal{Q}}$ into runs of $\widetilde{\mathcal{P}}$ with matching source and target. The harder part of the proof is to show the reciprocal statement: we consider the good runs of $\widetilde{\mathcal{P}}$ described in Lemma 18 and we show how to translate them, first into runs of $\widetilde{\mathcal{Q}}$, and finally into runs of $\overline{\mathcal{Q}}$.

Transforming runs of $\widetilde{\mathcal{Q}}$ and $\overline{\mathcal{Q}}$ into runs of $\widetilde{\mathcal{P}}$. The program $\widetilde{\mathcal{Q}}$ is a constrained version of $\widetilde{\mathcal{P}}$: every line is identical except for the lines with a popping instruction instead of a decrement, which are more restrictive since the correct symbol needs to be at the top of the stack. As a consequence, for every run $\widetilde{\pi}$ of $\widetilde{\mathcal{Q}}$ between two configurations x and y we immediately get a run π of $\widetilde{\mathcal{P}}$ between $\widetilde{h}(x)$ and $\widetilde{h}(y)$ which uses the same lines as $\widetilde{\pi}$.

Now given a run of $\overline{\pi}$ of $\overline{\mathcal{Q}}$ between two configurations x and y, translating $\overline{\pi}$ into a run of $\widetilde{\mathcal{P}}$ is not as direct since the lines 7–16 of $\overline{\mathcal{Q}}$ are not exactly analogous to the lines 7–8 of $\widetilde{\mathcal{P}}$. However, as we explained in the paper, using some local reshuffling $\widetilde{\mathcal{P}}$ can reproduce any counter update corresponding to the lines 7–16 of $\overline{\mathcal{Q}}$. This allows us to transform the run $\overline{\pi}$ into a run of $\widetilde{\mathcal{P}}$ between $\overline{h}(x)$ and $\overline{h}(y)$.

Transforming runs of $\widetilde{\mathcal{P}}$ into runs of $\widetilde{\mathcal{Q}}$. Let us suppose that $\mathcal{Q}$ delegates the counter b' but not c', and let π be a *good* run of $\widetilde{\mathcal{P}}$ as described in the proof of Lemma 18. We denote by x and y the starting and ending configuration of π. In order to transfer π to $\widetilde{\mathcal{Q}}$, we begin by creating an appropriate initial configuration x' as in the proof of Lemma 15. Formally, let $u_\pi \in \{\mathsf{b}', \mathsf{c}'\}^*$ be the word listing, in order, the occurrences of the decrements of b'' and c'' along π. We define the configuration x' of $\widetilde{\mathcal{Q}}$ by setting the values of the counters of X to the values they have in the starting configuration x of π, and setting the stack content to the reverse of the word u_π (so that the first letter of u_π is at the top of the stack). This definition guarantees that $h(x') = x$. To conclude, we need to argue that $\widetilde{\mathcal{Q}}$ can simulate π starting from x'. First, remark that the initialisation step is easily simulated since the definition of the initial stack content guarantees that the appropriate symbol is at the top of the stack whenever needed. We now explain, step by step, how $\widetilde{\mathcal{Q}}$ simulates the iteration steps of π. First, thanks to the definition of the initial stack content the loops on lines 5–7 can be iterated as in π. Then, we also iterate line 8 as in π. Remark that this disrupts the stack by adding a sequence of b' on top of it. Next comes the call to $\mathcal{P}$, and since π is a good run we know that this call is *correct*, in the sense that it starts in $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ (Equation (6)) and ends in $\textsc{Triple}(A \cdot 4^{B - \widetilde{F}(B)}, \widetilde{F}(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ (Equation (5)) for some $A, B \in \mathbb{N}_+$. Therefore in $\widetilde{\mathcal{Q}}$ we can simulate this call to $\mathcal{P}$ by a call to $\mathcal{Q}$, and since the value of b' is 0 in the ending configuration this implies that the call to $\mathcal{Q}$ will automatically pop all of the b' that were added on the stack. Therefore we are back with a stack content that matches a prefix of our initial stack content, and we can conclude the simulation of the iteration step by popping a single b'' from the stack.

Transforming runs of $\widetilde{\mathcal{P}}$ into runs of $\overline{\mathcal{Q}}$. Let us suppose that $\mathcal{Q}$ delegates both b' and c', and let π be a *good* run of $\widetilde{\mathcal{P}}$, as described in the proof of Lemma 18. We denote by x and y the starting and ending configuration of π. We show how to construct a run $\overline{\pi}$ of $\overline{\mathcal{Q}}$ that simulates π. First, remark that we have a single possibility for the starting configuration of $\overline{\pi}$: In the starting configuration of π only the values of a, b'' and c'' are nonzero (Equation (3)), and $\overline{\mathcal{Q}}$ only delegates b'' among these three counters. Therefore the initial stack content will just be a sequence of b'' of the appropriate length. Then, simulating the initialisation step of π is easy: one b'' is popped from the stack and the other counters are updated as in π.

To conclude, we show how to simulate the iteration steps visited by π. Let $\pi_1\pi_2\pi_3$ be a subrun of π corresponding to an iteration step of $\widetilde{\mathcal{P}}$, where π_2 stands for the call to the program $\mathcal{P}$. As we showed in the proof of Lemma 18, every call to $\mathcal{P}$ along π is *correct*, in the sense that it starts in some configuration $\textsc{Triple}(A, B, \mathsf{a}, \mathsf{b}', \mathsf{c}', \mathsf{X})$ (Equation (6)) and ends in $\textsc{Triple}(A \cdot 4^{B - \widetilde{F}(B)}, \widetilde{F}(B), \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{X})$ for some $A, B \in \mathbb{N}_+$ (Equation (5)). As a consequence, since $\mathcal{Q}$ simulates $\mathcal{P}$, there exists a run $\overline{\pi}_2$ of $\mathcal{Q}$ that simulates π_2, but this run requires a starting stack content corresponding to some specific shuffle u_{π_2} of the word $(\mathsf{b}')^{\mathsf{B}}(\mathsf{c}')^{\mathsf{A}\cdot(4^{\mathsf{B}}-1)}$. Fortunately, as we explained in the paper, the lines 7–16 of $\overline{\mathcal{Q}}$ allow to push any shuffle of b' and c' on the stack. In particular, there exists a subrun $\overline{\pi}_1$ of $\overline{\mathcal{Q}}$ that simulates π_1 and pushes the word u_{π_2} on the stack. As a consequence, $\overline{\pi}_1\overline{\pi}_2$ simulates truthfully the subrun $\pi_1\pi_2$ with no impact on the stack: $\overline{\pi}_1$ pushes u_{π_2}, which is then popped by $\overline{\pi}_2$. Therefore, we can simulate the iteration step $\pi_1\pi_2\pi_3$ by starting with $\overline{\pi}_1\overline{\pi}_2$, and then adding $\overline{\pi}_3$ which pops a single b from the stack to simulate the decrement of b occurring in π_3.

Approximately Interpolating Between Uniformly and Non-Uniformly Polynomial Kernels

Akanksha Agrawal
Indian Institute of Technology Madras, India

M. S. Ramanujan
University of Warwick, Coventry, UK

Abstract

The problem of computing a minimum set of vertices intersecting a finite set of forbidden minors in a given graph is a fundamental graph problem in the area of kernelization with numerous well-studied special cases.

A major breakthrough in this line of research was made by Fomin et al. [FOCS 2012], who showed that the ρ-Treewidth Modulator problem (delete minimum number of vertices to ensure that treewidth is at most ρ) has a polynomial kernel of size $k^{g(\rho)}$ for some function g. A second standout result in this line is that of Giannapoulou et al. [ACM TALG 2017], who obtained an $f(\eta)k^{\mathcal{O}(1)}$-size kernel (for some function f) for the η-Treedepth Modulator problem (delete fewest number of vertices to make treedepth at most η) and showed that some dependence of the exponent of k on ρ in the result of Fomin et al. for the ρ-Treewidth Modulator problem is unavoidable under reasonable complexity hypotheses.

In this work, we provide an approximate interpolation between these two results by giving, for every $\epsilon > 0$, a $(1+\epsilon)$-approximate kernel of size $f'(\eta, \rho, 1/\varepsilon) \cdot k^{g'(\rho)}$ (for some functions f' and g') for the problem of deciding whether k vertices can be deleted from a given graph to obtain a graph that has elimination distance at most η to the class of graphs that have treewidth at most ρ.

Graphs of treedepth η are precisely the graphs with elimination distance at most $\eta - 1$ to the graphs of treewidth 0 and graphs of treewidth ρ are simply graphs with elimination distance 0 to graphs of treewidth ρ. Consequently, our result "approximately" interpolates between these two major results in this active line of research.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Lossy Kernelization, Treewidth Modulator, Vertex Deletion Problems

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.36

Funding Akanksha Agrawal is partly supported by Science and Engineering Research Board, Startup Grant (SRG/2022/000962), and Veena and Induprakas Keri Faculty Fellowship. M. S. Ramanujan is supported by Engineering and Physical Sciences Research Council (EPSRC) grants EP/V007793/1 and EP/V044621/1.

1 Introduction

Polynomial-time preprocessing is one of the widely used methods to tackle NP-hardness in practice, and the area of *kernelization* has been extremely successful in laying down a mathematical framework for the design and rigorous analysis of preprocessing algorithms for decision problems. The central notion in this area is that of a *kernelization* algorithm (whose output is often called a *kernel*), which is a preprocessing algorithm that runs in polynomial time and transforms a "large" instance of a decision problem into a significantly smaller, but equivalent instance. Over the last decade, the area of kernelization has seen the development of a wide range of tools to design preprocessing algorithms and lower bounds techniques. The reader may find an introduction to the field in [16, 18, 6, 8]. An "efficient

© Akanksha Agrawal and M. S. Ramanujan;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 36; pp. 36:1–36:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

preprocessing algorithm" in this setting is referred to as a *polynomial kernelization* and is simply a kernelization algorithm whose output has size bounded polynomially in a parameter of the input. The central classification task in the area of kernelization is to identify NP-hard problems and associated parameters for which polynomial kernels exist.

One of the most intensively investigated problems in the literature on kernelization is the $\mathcal{F}$-Deletion problem and its special cases, which are well-studied NP-complete problems in their own right. In this problem, $\mathcal{F}$ is a fixed finite family of graphs and one is given a graph G and an integer k as input. The objective is to determine whether at most k vertices can be deleted from G so that the resulting graph is $\mathcal{F}$-minor free (does not contain a minor isomorphic to a graph in $\mathcal{F}$). Well-studied special cases of this problem include Vertex Cover ($\mathcal{F} = \{K_2\}$), Feedback Vertex Set ($\mathcal{F} = \{K_3\}$), Planarization ($\mathcal{F} = \{K_{3,3}, K_5\}$) [21], Diamond Hitting Set ($\mathcal{F} = \{\theta_3\}$) [9], Pathwidth One Vertex Deletion ($\mathcal{F} = \{K_3, T_2\}$) [22], and θ_c-Deletion [14, 10].

A common feature shared by many such well explored special cases of this problem is that $\mathcal{F}$ contains at least one planar graph. Motivated by this, Fomin et al. [11] in an influential work, investigated this restriction of the problem (when the family $\mathcal{F}$ contains at least one planar graph) and demonstrated the existence of a polynomial kernel for every such $\mathcal{F}$. This particular variant of $\mathcal{F}$-Deletion is known in the literature as the Planar $\mathcal{F}$-Deletion problem. More precisely, Fomin et al. showed that for every such family $\mathcal{F}$, there is a function g depending only on $\mathcal{F}$ such that Planar $\mathcal{F}$-Deletion has a polynomial kernel of size bounded by $k^{g(\mathcal{F})}$. This raised the question of designing a *uniformly-polynomial kernel*, i.e., a kernel of size at most $f(\mathcal{F}) \cdot k^c$ for some universal constant c that does not depend on $\mathcal{F}$. Subsequently, Giannopoulou [12] addressed this question and proved that: (i) even when the $\mathcal{F}$-minor free graphs under consideration are the graphs of treewidth at most a constant ρ, one does not expect a uniformly-polynomial kernel for Planar $\mathcal{F}$-Deletion (this problem is called ρ-Treewidth Modulator) unless $\mathsf{NP} \subseteq$ co-NP.), and (ii) if the $\mathcal{F}$-minor free graphs correspond to the class of graphs of *treedepth* at most some constant η (i.e., for the η-Treedepth Modulator problem), then there is indeed a kernel of size $f(\eta) \cdot k^c$ for some constant c independent of η, i.e., there is a uniformly-polynomial kernel. In fact, they obtain a kernel of size $2^{\mathcal{O}(\eta^2)} \cdot k^c$. Note that the treedepth of a graph expresses the number of rounds needed to obtain an empty graph by removing one vertex from every connected component in each round and this parameter upper bounds treewidth.

1.1 Our work

The aforementioned results of Fomin et al. [11] and Giannopoulou et al. [12] are the starting point of this work. Our main goal is to obtain a unifying result "interpolating" between the positive results of both these papers. That is, we would like to obtain an algorithm where, by instantiating various parameters appropriately we can derive both their results as special cases (at least in the qualitative sense contrasting uniformly vs non-uniformly polynomial kernels, i.e., ignoring the precise growth rate of the functions f and g). However, at first glance, it is not even clear what problem such an algorithm could be required to solve. Our first contribution therefore is to introduce a new problem (which we call Deletion (η/ρ)-Elimination) as the ideal candidate for this task. This problem is built on the recently popular notion of elimination distance [3, 4, 1, 17, 13] and is formally described as follows: The input is a graph G, and integer k and the goal is to decide whether there is a set $S \subseteq V(G)$ of size at most k such that $G - S$ has elimination distance at most η to graphs of treewidth ρ?

The notion of elimination distance of a graph G to a graph class $\mathcal{H}$ was introduced by Bulian and Dawar [3] and roughly speaking, it expresses the number of rounds needed to obtain a graph in $\mathcal{H}$ by removing one vertex from every connected component in each round. Notice how elimination distance naturally generalizes treedepth. We refer the reader to Section 2 for formal definitions of these notions. At this point, it is sufficient to know that:

- Graphs of treedepth η are precisely the graphs with elimination distance at most $\eta - 1$ to the graphs of treewidth 0. Therefore, when $\rho = 0$, Deletion (η/ρ)-Elimination is nothing but the $(\eta + 1)$-Treedepth Modulator problem.
- Graphs of treewidth ρ are simply graphs with elimination distance 0 to graphs of treewidth ρ. Therefore, when $\eta = 0$, Deletion (η/ρ)-Elimination is simply the ρ-Treewidth Modulator problem.

Thus, if one were to obtain a kernel of size $f(\eta, \rho) \cdot k^{g(\rho)}$ (where the function g is independent of η) for the Deletion (η/ρ)-Elimination problem, then by setting $\eta = 0$ one would obtain the non-uniformly polynomial kernel result of Fomin et al., whereas setting $\rho = 0$ gives the uniformly-polynomial kernel result of Giannopolou et al. [12] (again, ignoring the precise growth rates of f and g). Although we have not settled this question completely in this paper, we provide an *approximate polynomial kernel* [20] for this problem whose error can be made arbitrarily small thus providing an arbitrarily refined interpolation between these two results. Stated formally, we prove the following:

▶ **Theorem 1.** *For every $\rho, \eta \in \mathbb{N}$ and $0 < \varepsilon < 1$, (ρ/η)-Modulator has a $(1 + \varepsilon)$-approximate kernel of size $f(\eta, \rho, 1/\varepsilon) \cdot k^{g(\rho)}$ for some functions f and g.*

Approximate kernels are a useful relaxation of the standard notion of kernels and combine well with approximation algorithms. Informally speaking, an α-approximate kernel is a polynomial-time algorithm that given as input a pair (I, k) where I is the problem instance and k is the parameter, outputs an instance (I', k') of the same problem such that $|I'| + k' \leq g(k)$ for some computable function g and any c-approximate solution for the instance I' can be turned in polynomial time into a $(c \cdot \alpha)$-approximate solution for the original instance I. We refer the reader to Section 2 for a formal definition of all related terms.

By instantiating $\eta = 0$ and $\rho = 0$ respectively in Theorem 1, we obtain, for every $0 < \varepsilon < 1$, a $(1 + \varepsilon)$-approximate kernel of size at most $f_1(\rho, 1/\varepsilon)k^{g(\rho)}$ for ρ-Treewidth Modulator for some functions f_1 and g, and a $(1 + \varepsilon)$-approximate kernel of size at most $f_2(\eta, 1/\varepsilon)k^c$ for ρ-Treewidth Modulator for some function f_2 and a universal constant c independent of η and ε.

2 Preliminaries

We refer to the book of Diestel [7] for standard graph terminology. Whenever the context is clear, we use n and m to denote the number of vertices and the number of edges in the input graph, respectively. The set $\mathcal{C}(G)$ denotes the set of connected components of G. For sets $X, Y \subseteq V(G)$, an X-Y *separator* in G is a set $S \subseteq V(G)$, such that $G - S$ has no $x - y$ path, where $x \in X \setminus S$ and $y \in Y \setminus S$. By $\mathsf{sep}_G(X, Y)$ we denote the size of a minimum sized X-Y separator in G. For a tree T and vertices $u, v \in V(T)$, we denote the unique path between u and v by $\mathsf{Pth}_T(u, v)$. A *rooted tree* is a tree with a special vertex called the *root* of the tree. Consider a rooted tree T with root r. A vertex $t \in V(T) \setminus \{r\}$ is a *leaf* of T if it is a vertex of degree exactly one in T. Moreover, if $V(T) = \{r\}$, then r is the leaf (as well as the root) of T. A vertex which is not a leaf, is a *non-leaf* vertex. By $\mathsf{desc}_T(t)$, we denote the set of all descendants of t in T. The subscript T is dropped when clear from the context.

A *rooted forest* is a forest where each of its connected component is a rooted tree. For a rooted forest F, a vertex $v \in V(F)$ that is not a root of any of its rooted trees is a *leaf* if it is of degree exactly one in F. We denote the set of leaves in a rooted forest by $\mathsf{Lf}(F)$. The *depth*, denoted by $\mathsf{depth}(T)$ of a rooted tree T is the maximum number vertices in a root to leaf path in T. The *depth*, denoted by $\mathsf{depth}(F)$ of a rooted forest is the maximum over the depths of its rooted trees.

Least Common Ancestor-Closure of Sets in Trees. For a rooted tree T and vertex set $M \subseteq V(T)$ the least common ancestor-closure (*LCA-closure*), denoted by $\mathsf{LCA\text{-}closure}_T(M)$, is obtained by the following process. Initially, set $M' = M$. Then, as long as there are vertices x and y in M' whose least common ancestor w is not in M', add w to M'. When the process terminates, output M' as the LCA-closure of M. The following folklore result summarizes the properties of LCA-closures which we will use in this paper.

▶ **Observation 2.** *Let T be a rooted tree, $M \subseteq V(T)$ and $M' = \mathsf{LCA\text{-}closure}_T(M)$. Then $|M'| \leq 2|M|$ and for every connected component C of $T - M'$, $|N(C)| \leq 2$. Moreover, the number of components of $T - M'$ that have exactly 2 neighbors in M' is at most $|M'| - 1$.*

Tree decompositions. A *tree decomposition* of a graph G is a pair (T, β), where T is a tree rooted at r and $\beta : V(T) \to 2^{V(G)}$ that satisfies the following properties: (i) $\bigcup_{t \in V(T)} \beta(t) = V(G)$, (ii) for every edge $\{u, v\} \in E(G)$ there is a node $t \in V(T)$, such that $u, v \in \beta(t)$, and (iii) for every $v \in V(G)$, the graph $T[X_v]$ is a subtree of T, where $X_v = \{t \in V(T) \mid v \in \beta(t)\}$. For $t \in V(T)$, we call $\beta(t)$ the *bag* of t. The sets in $\{\beta(t) \mid t \in V(T)\}$ are called *bags* of (T, β). We refer to the vertices in $V(T)$ as *nodes*. The *width* of the tree decomposition (T, β) is $\max_{t \in V(T)} |\beta(t)| - 1$. The *treewidth* of G, denoted by $\mathsf{tw}(G)$, is the minimum over the widths over all possible tree decompositions of G. For $\rho \in \mathbb{N}$, by $\mathcal{T}_\rho$, we denote the family of graphs of treewidth bounded by ρ.

▶ **Proposition 3.** *Consider a graph G, a tree decomposition (T, β) of G, and a clique $S \subseteq V(G)$ in G. Then, there is $t \in V(T)$, such that $S \subseteq \beta(t)$.*

▶ **Proposition 4.** *For a graph G and distinct non-adjacent vertices $u, v \in V(G)$ such that $\mathsf{sep}_G(\{u\}, \{v\}) \geq \mathsf{tw}(G) + 2$, we have $\mathsf{tw}(G) = \mathsf{tw}(G + \{(u, v)\})$. That is, adding the edge (u, v) does not increase the treewidth of G.*

▶ **Proposition 5.** *Consider an integer ρ and two graphs H_1 and H_2, where:*
1. *$Z = V(H_1) \cap V(H_2)$ is a clique in both H_1 and H_2, and*
2. *$\mathsf{tw}(H_1), \mathsf{tw}(H_2) \leq \rho$.*

Then, $\mathsf{tw}(H_1 \cup H_2) \leq \rho$, where $H_1 \cup H_2$ is the graph with $V(H_1 \cup H_2) = V(H_1) \cup V(H_2)$ and $E(H_1 \cup H_2) = E(H_1) \cup E(H_2)$.

▶ **Proposition 6** ([15]). *For a graph G, in time $2^{\mathcal{O}(\mathsf{tw}(G))}|V(G)|$, we can compute a tree decomposition of G of width at most $2 \cdot \mathsf{tw}(G) + 1$.*

Forest embeddings and $(\eta, \mathcal{H})$-decompositions. We define some useful definitions and give some preliminary results regarding forest embeddings and $(\eta, \mathcal{H})$-decompositions.

▶ **Definition 7** (Forest embedding). A *forest embedding* of a graph G is a pair (F, f), where F is a rooted forest and $f : V(G) \to V(F)$ is a bijective function, such that for each $\{u, v\} \in E(G)$, either $f(u)$ is a descendant of $f(v)$, or $f(v)$ is a descendant of $f(u)$. The

depth of the forest embedding (F, f) is the depth of the rooted forest F.[1] The *treedepth* of a graph G, denoted by $\mathsf{td}(G)$, is the minimum over the depths over all possible forest embeddings of G.

The next observation states that for every clique S in G and every forest embedding of G, there is a root-to-leaf path in this forest embedding that contains all the vertices of S.

▶ **Observation 8.** *Consider a graph G, a forest embedding $f : V(G) \to V(F)$ of G into the rooted forest F, and a clique $S \subseteq V(G)$ in G. Then, there is a rooted tree $T \in \mathcal{C}(F)$[2] with root r and a leaf $t \in V(T)$, such that for each $s \in S$, we have $f(s) \in V(\mathsf{Pth}_T(r, t))$. That is, the vertices in S must be mapped to vertices in one root-to-leaf path in a tree of F.*

Next, we recall the notion of *elimination distance* introduced by Bulian and Dawar [3]. We rephrase their definition and introduce notation that will facilitate our presentation in this paper.

▶ **Definition 9** (Elimination Distance and $(\eta, \mathcal{H})$-decompositions)**.** Consider a family $\mathcal{H}$ of graphs and an integer $\eta \in \mathbb{N}$. An $(\eta, \mathcal{H})$-decomposition of a graph G is a tuple $(X, Y, F, f : X \to V(F), g : \mathcal{C}(G[Y]) \to \mathsf{Lf}(F) \cup \{\perp\})$, where (X, Y) is a partition of $V(G)$ and F is a rooted forest of (at most) depth η, such that the following conditions are satisfied:

1. (F, f) is a forest embedding of $G[X]$,
2. each connected component of $G[Y]$ belongs to $\mathcal{H}$, and
3. for a connected component C of $G[Y]$, a vertex $v \in V(C)$, and an edge $\{u, v\} \in E(G)$, either $u \in Y$ or $f(u)$ is a vertex in the unique path in F from r to $g(C)$, where r is the root of the connected component in F containing the vertex $g(C)$.[3]

We say that G admits an $(\eta', \mathcal{H})$-decomposition if there is some $\eta \leq \eta'$, for which there is an $(\eta, \mathcal{H})$-decomposition of G. The *elimination distance* of G to $\mathcal{H}$ (or the $\mathcal{H}$-*elimination distance* of G) is the smallest integer η^* for which G admits an $(\eta^*, \mathcal{H})$-decomposition.

Note that when $\mathcal{H}$ is the class of edgeless graphs, then a graph has elimination distance η to $\mathcal{H}$ precisely when it has treedepth at most η. On the other hand, for every class $\mathcal{H}$, a graph G has elimination distance 0 to $\mathcal{H}$ precisely when $G \in \mathcal{H}$. A graph has elimination distance 0 to graphs of treewidth at most ρ if and only if it has treewidth at most ρ.

Consider an $(\eta, \mathcal{H})$-decomposition $\mathbb{D} = (X, Y, F, f, g)$ of a graph G. We say that X is the *interior part* of $\mathbb{D}$ and Y is the *exterior part* of $\mathbb{D}$. For a leaf $u \in \mathsf{Lf}(F)$, by $\widehat{P}_u^{\mathbb{D}}$ we denote the path from u to r in the tree T, where T is the tree rooted at r in F, containing u. Moreover, by $P_u^{\mathbb{D}}$, we denote the graph $G[\{f^{-1}(w) \mid w \in V(\widehat{P}_u^{\mathbb{D}})\}]$. For a connected component $C \in \mathcal{C}(G[Y])$, by $C_{\mathsf{ext}}^{\mathbb{D}}$ we denote the graph $G[V(C) \cup \{f^{-1}(w) \mid w \in V(\mathsf{Pth}_F(g(C), r))\}]$, where r is the root of the component of F containing $g(C)$. (For the above notations we drop the superscript $\mathbb{D}$, when it is clear from the context.)

▶ **Observation 10.** *If (X, Y, F, f, g) is an $(\eta, \mathcal{H})$-decomposition of a graph G such that $|V(G)| \geq 2$, then there is an $(\eta', \mathcal{H})$-decomposition (X', Y', F', f', g') such that (1) F' is a rooted tree with at least two vertices, (2) $f' : X \to V(F')$ is a bijective function, (3) $g' : \mathcal{C}(G[Y]) \to \mathsf{Lf}(F')$, and (4) $\eta' = \mathsf{depth}(F') \leq \eta + 2$.*

[1] Sometimes we slightly abuse the notation for simplicity, and say that, for every $\beta \geq \alpha$, (F, f) is a forest embedding of depth β, where α is the depth of F.
[2] Recall that $\mathcal{C}(F)$ denotes the set of connected components of F,
[3] If $g(C) = \perp$, then u must belong to Y.

For a graph G and integers $\eta, \rho \in \mathbb{N}$, a set $S \subseteq V(G)$ is a (ρ/η)-modulator, if $G - S$ admits an $(\eta, \mathcal{T}_\rho)$-decomposition. Next we state a pair of results regarding the computation of $(\eta, \mathcal{T}_\rho)$-decompositions in bounded treewidth graphs that use Courcelle's theorem [5] and the Counting Monadic Second Order Logic (CMSO)-expressibility of graphs of treewidth ρ as well as that of elimination distance to CMSO-expressible graph classes.

▶ **Observation 11.** *There is an algorithm that, given a graph G and non-negative integers ρ and η, runs in time $f(\rho, \eta, \mathsf{tw}(G))n^{\mathcal{O}(1)}$ for some computable function f and computes a minimum (ρ/η)-modulator S of G and an $(\eta, \mathcal{T}_\rho)$-decomposition of $G - S$.*

We note that in order to obtain an $(\eta, \mathcal{T}_\rho)$-decomposition of $G - S$ given S, one cannot use a direct invocation of Courcelle's theorem on $G - S$. However, one can use standard self-reducibility arguments using repeated invocations of Courcelle's theorem. In particular, if we can identify a vertex v (which must exist) such that $G - S - \{v\}$ has a $(\eta - 1, \mathcal{T}_\rho)$-decomposition, then we can place v at the "top" of the decomposition, delete it and recurse into each connected component.

▶ **Observation 12.** *Given a graph H, a vertex set Q in H, and integers η' and ρ', in time $\widehat{h}(\eta', \rho', \mathsf{tw}(H)) \cdot |V(H)|^{\mathcal{O}(1)}$, we can correctly do one of the following: i) determine the existence of an $(\eta', \mathcal{T}_{\rho'})$-decomposition $\mathbb{D} = (X, Y, F, f, g)$ of H, such that $Q \subseteq Y$, or ii) conclude that such a decomposition does not exists.*

Parameterized problems and (approximate) kernels. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ. An instance of a parameterized problem consists of a pair (x, k), where k is called the *parameter*. We assume that k is *given in unary* and hence $k \leq |x|$.

▶ **Definition 13** (Kernelization)**.** Let $\Pi \subseteq \Gamma^* \times \mathbb{N}$ be a parameterized problem and g be a computable function. We say that Π *admits a kernel of size g* if there exists an algorithm referred to as a *kernelization* algorithm (or a *kernel*) that, given $(x, k) \in \Gamma^* \times \mathbb{N}$, outputs in time polynomial in $|x| + k$, a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$, and (b) $|x'| + k' \leq g(k)$.

▶ **Definition 14** ([20])**.** A parameterized optimization (minimization or maximization) problem is a computable function $\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$.

The *instances* of a parameterized optimization problem Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$, and a *solution* to (I, k) is simply a string $s \in \Sigma^*$, such that $|s| \leq |I| + k$. The *value* of the solution s is $\Pi(I, k, s)$. Since we only deal with a minimization problem in this work, we state some of the definitions only in terms of minimization problems when the definition for maximization problems is analogous. The parameterized optimization version of Deletion (η/ρ)-Elimination is a minimization problem with the optimization function ρ/η-Del : $\Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\infty\}$ defined as follows.

$$\rho/\eta\text{-}\mathsf{Del}(G, k, S) = \begin{cases} \infty & \text{if } S \text{ is not a solution for } G, \\ \min\{|S|, k+1\} & \text{otherwise.} \end{cases}$$

▶ **Definition 15** ([20])**.** For a parameterized minimization problem Π, the *optimum value* of an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ is $\mathsf{OPT}_\Pi(I, k) = \min_{\substack{s \in \Sigma^* \\ |s| \leq |I| + k}} \Pi(I, k, s)$.

Consequently, for Deletion (η/ρ)-Elimination, we define $\mathsf{OPT}(G, k) = \min_{S \subseteq V(G)} \rho/\eta\text{-}\mathsf{Del}\ (G, k, S)$.

In the above definition, k is effectively a threshold; for solutions of size at most k we care about what their size is, while all solutions of size larger than k are *equally bad*, and are consequently assigned value $k+1$. We point the interested reader to Section 2.1, [20] and Section 3.2, [19] for an in-depth discussion of these definitions and their motivations.

We now recall the other relevant definitions from [20] regarding *approximate kernels*.

▶ **Definition 16** ([20]). Let $\alpha \geq 1$ be a real number and Π be a parameterized minimization problem. An *α-approximate polynomial time preprocessing algorithm* $\mathcal{A}$ for Π is a pair of polynomial-time algorithms. The first one is called the *reduction algorithm*, and computes a map $\mathcal{R}_{\mathcal{A}} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$. Given as input an instance (I, k) of Π, the reduction algorithm outputs another instance $(I', k') = \mathcal{R}_{\mathcal{A}}(I, k)$.

The second algorithm is called the *solution-lifting algorithm*. This algorithm takes as input an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ of Π, the output instance (I', k') of the reduction algorithm, and a solution s' to the instance (I', k'). The solution-lifting algorithm works in time polynomial in $|I|$,k,$|I'|$,k' and s', and outputs a solution s to (I, k) such that $\frac{\Pi(I,k,s)}{\mathsf{OPT}(I,k)} \leq \alpha \cdot \frac{\Pi(I',k',s')}{\mathsf{OPT}(I',k')}$.

The *size* of a polynomial time preprocessing algorithm $\mathcal{A}$ is a function $\mathsf{size}_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ defined as $\mathsf{size}_{\mathcal{A}}(k) = \sup\{|I'| + k' : (I', k') = \mathcal{R}_{\mathcal{A}}(I, k), I \in \Sigma^*\}$.

▶ **Definition 17** (α-approximate kernelization). An *α-approximate kernelization* (or *α-approximate kernel*) for a parameterized optimization problem Π, and real $\alpha \geq 1$, is an α-approximate polynomial time preprocessing algorithm $\mathcal{A}$ for Π such that $\mathsf{size}_{\mathcal{A}}$ is upper bounded by a computable function $g : \mathbb{N} \to \mathbb{N}$. We say that $\mathcal{A}$ is an α-approximate polynomial kernelization if g is a polynomial function.

▶ **Definition 18** ([20]). Let $\alpha \geq 1$ be a real number, and Π be a parameterized minimization problem. An α-approximate polynomial time preprocessing algorithm for Π is said to be *strict* if, for every instance (I, k), reduced instance $(I', k') = \mathcal{R}_{\mathcal{A}}(I, k)$ and solution s' to (I', k'), the solution s to (I, k) output by the solution-lifting algorithm when given s' as input satisfies the following.

$$\frac{\Pi(I, k, s)}{OPT(I, k)} \leq \max\left\{\frac{\Pi(I', k', s')}{OPT(I', k')}, \alpha\right\}$$

The notion of *strictness* in the above direction allows one to "chain" multiple α-approximate preprocessing algorithms to obtain a single α-approximate preprocessing algorithm.

▶ **Definition 19.** A *reduction rule* is simply the reduction algorithm of a polynomial time preprocessing algorithm. The reduction rule *applies* if the output instance of the reduction algorithm is not the same as the input instance.

▶ **Definition 20** ([20]). A reduction rule is said to be *α-safe for* Π if it is the reduction algorithm of a strict α-approximate polynomial time preprocessing algorithm for Π. A reduction rule is *safe* if it is 1-safe.

In the context of our work, it is sufficient for us to argue that all our reduction rules are $(1 + \varepsilon)$-safe.

▶ **Definition 21** ([20]). A *polynomial-size approximate kernelization scheme* (PSAKS) for a parameterized optimization problem Π is a family of α-approximate polynomial kernelization algorithms, with one such algorithm for every $\alpha > 1$.

3 The PSAKS for Deletion (η/ρ)-Elimination

3.1 Overview of our algorithm

We begin with an overview of our algorithm. Let (G, k) be the given instance of Deletion (η/ρ)-Elimination. We begin by computing a factor-$\mathcal{O}(1)$-approximate solution S for G (see Lemma 22 in Section 3.2). That is, there is a constant γ such that S is a (ρ/η)-modulator of size at most $\gamma \cdot \mathsf{OPT}$. If $|S| > \gamma k$, then we conclude that the input instance is a no-instance and so, we return a trivial no-instance of size $\mathcal{O}(k)$. Henceforth, we assume that $|S| = \mathcal{O}(k)$. We next compute an $(\eta, \mathcal{T}_\rho)$-decomposition (X, Y, F, f, g) of $G - S$ in polynomial time using Observation 11. By Observation 10, we may assume without loss of generality that (X, Y, F, f, g) is an $(\eta', \mathcal{T}_\rho)$-decomposition of $G - S$ where F is a tree and $\eta' \leq \eta + 2$.

We next execute a series of reduction rules that either delete vertices or add edges between non-adjacent vertices. Each such rule reduces our instance complexity by a polynomial (in n) amount after each application, where the rules are applied in the order in which we state them. In particular, assuming n denotes $|V(G)|$, our algorithm will apply a total of at most $\binom{n}{2} + n$ reduction rules. Moreover each of these rules is either 1-safe or $(1 + \varepsilon)$-safe (see Definition 20), implying that by combining them, we get the required PSAKS.

We now proceed to sketch each reduction rule and give intuitive reasonings behind their correctness. The formal description of the rules and correctness arguments can be found in Section 3.2.

3.1.1 Overview of the reduction rules

Rule 1: Identifying highly-connected pairs. Our first reduction rule adds edges between non-adjacent pairs of vertices in G, that have sufficiently high flow between them. This reduction rule is based on the fact that, roughly speaking, for such pairs of non-adjacent vertices u, v one of the following holds with respect to some solution S^* (i.e., a (ρ/η)-modulator of size at most k) and an $(\eta, \mathcal{T}_\rho)$-decomposition of $G - S^*$, call it $\mathbb{D}$:

- **(i)** At least one of them belongs to S^*.
- **(ii)** Both belong to the interior of $\mathbb{D}$ and are mapped to vertices in a single root-to-leaf path in the associated forest.
- **(iii)** Exactly one of them belongs to a connected component C in the exterior of $\mathbb{D}$ while the other belongs to the interior of the same decomposition and is mapped to a vertex of the associated forest, which is an ancestor of the leaf-vertex with which C is associated.
- **(iv)** Both belong to a bag in any optimal tree decomposition of the exterior part of $\mathbb{D}$.

In any of these cases, the addition of such an edge (u, v) maintains equivalence with the given instance. A similar observation was crucially used in the work of Fomin et al. [11] and this rule (adding edges between highly connected pairs of non-adjacent vertices) was also explicitly used by Giannapolou et al. [12] in their work. The reduction in the instance complexity given by this rule is captured by the fact that the number of non-edges strictly decreases with each application.

Rule 2: Bounding the degree of F (and hence, the size of the interior X) and the number of connected components in $G - (S \cup X \cup \mathsf{Mark}_1)$ where X and F are given by the current decomposition (X, Y, F, f, g) and Mark_1 is a marked set of vertices to be defined. Note that since the depth of F is bounded by η', if we have a bound on the degree of F, then we can obtain a bound on the number of vertices in X. This part of our work is also inspired by arguments used by Fomin et al. [11] and Giannopolou et al. [12] (the latter in the setting of treedepth modulators).

Towards designing rules that help us achieve the required degree bound, we begin by employing the *near-protrusion decomposition* technique of Fomin et al. [11] as follows. We first create a marked set of vertices, denoted by Mark_1, in $G-S$. We do this by adding a minimum separator to our marked set, for every pair of non-adjacent vertices in S. We remark that inapplicability of our first reduction rule ensures that the sizes of such separators are "small". For technical reasons, we will in fact add more vertices to Mark_1 besides what we discussed above and we defer the precise description of these vertices to the next subsection. After we remove $S \cup \mathsf{Mark}_1$ from G, we obtain a set of connected components. We observe that the first reduction rule ensures that the neighborhood of each such connected component inside S is a clique. Moreover, we observe that the neighborhood of each such connected component disjoint from S has size bounded by $\mathcal{O}(\rho+\eta)$. We next analyze these connected components and group them into different classes based on an appropriate notion of equivalence, which depends on the following for each such connected component C:

- Firstly, we have a set $\widehat{S}'$ of at most $\rho+1$ vertices from $S \cap N(C)$. Intuitively speaking $\widehat{S}'$ represents the vertices from S that are not deleted in some hypothetical optimal solution and instead, lie in the exterior part of the associated optimal decomposition (since $S \cap N(C)$ is shown to be a clique it follows that these vertices must be contained in a single bag of any tree-decomposition of the exterior part).
- The second element determining the equivalence class of C will be a set $B \subseteq X \cap N(C)$ of size at most $\eta+1$. This set denotes the vertices in $X \cap N(C)$ that are disjoint from an optimal solution.
- We will further partition the above set B into two sets, B_1 and B_2, that will correspond to the subsets that go to the interior and exterior of an optimal decomposition, respectively.
- Finally we have an integer $\widehat{\eta}$ which will denote the minimum integer for which an $(\widehat{\eta}, \mathcal{T}_\rho)$-decomposition for the connected component exists, such that $\widehat{S}' \cup B_2$ belong to the exterior of this decomposition.

Once we formalize the above equivalence, we show that (a) the number of possible equivalence classes to which C can belong is bounded polynomially in k and (b) if, for the equivalence class containing C we have sufficiently many other connected components, then we can safely reduce the size of the instance by deleting C. Once this is done, we obtain bounds on: i) the number of vertices in X, and ii) the number of connected components in $G-(S \cup X \cup \mathsf{Mark}_1)$.

From this point onwards, in order to obtain our approximate kernel, it is enough to bound the sizes of the connected components in $G-(S \cup X \cup \mathsf{Mark}_1)$. To achieve this, we will in fact bound the sizes of the connected components in $G-(S \cup \mathsf{Mark}_1)$. Towards this, we design our next two reduction rules, out of which the next rule is the only "lossy" reduction rule in our algorithm.

Rule 3: Converting near protrusions to protrusions. Using the terminology of Fomin et al., the connected components of $G-(S \cup \mathsf{Mark}_1)$ are *near protrusions*. Essentially, for each connected component C of $G-(S \cup \mathsf{Mark}_1)$, the graph induced on C has bounded treewidth (in this case, treewidth $O(\rho+\eta)$) and moreover, any (ρ,η)-modulator of size at most k contains all but at most $\kappa(\rho+\eta)$ neighbors of C for some constant κ. We note that in standard terminology, such a connected component C would be a $\mathcal{O}(\rho+\eta)$-protrusion if the neighborhood of C is also bounded by $\mathcal{O}(\rho+\eta)$, in which case one can rely on several "protrusion-reduction" techniques in the literature. However, dealing with near protrusions requires a more careful approach because there could be $k^{\mathcal{O}(\rho+\eta)}$ possible ways in which an optimal solution intersects $N(C)$. As our main aim is to avoid the dependence on η in the exponent of k, this obstacle cannot be overcome only by using techniques from literature

and requires a new approach. The key insight in this part of our work lies in a reduction rule that enables us to delete vertices in such a way that we eliminate near protrusions that are not already $\mathcal{O}(1/\varepsilon(\rho+\eta))$-protrusions. However, this rule is not lossless although the loss ε can be made arbitrarily close to 0 (at a cost of larger kernel size eventually). We next describe the key insight that facilitates this.

Consider a connected component C of $G-(S\cup\mathsf{Mark}_1)$ such that $N(C) > (\kappa/\varepsilon+1)\cdot(\rho+\eta)$ where κ is the constant described in the previous paragraph. Then, we have that by greedily adding $N(C)$ to any solution we eventually compute (i.e., by deleting $N(C)$ from the graph and modifying the budget k accordingly), we will correctly pick at least $(\kappa/\varepsilon)\cdot(\rho+\eta)$ vertices of an optimal solution and incorrectly pick at most an ε fraction of $|N(C)|$ to the solution that is eventually returned. This is the informal reasoning behind the $(1+\varepsilon)$-safeness of this rule. By repeating this rule exhaustively, we arrive at an instance where every component of $G-(S\cup\mathsf{Mark}_1)$ is a $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusion, i.e., it induces a graph with treewidth at most $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$ and has a boundary of size at most $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$ through which it interacts with the rest of the graph.

Rule 4: Bounding the size of components in $G-(S\cup\mathsf{Mark}_1)$, i.e., protrusion reduction. Here, we use the "lossless protrusion replacer" of Fomin et al. [11] to reduce all remaining $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusions to a size that depends only on ε,ρ and η. Essentially, this is a subroutine that replaces any sufficiently large $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusion in $G-(S\cup\mathsf{Mark}_1)$ with a strictly smaller one, repeatedly. In the process, one is guaranteed that any feasible solution for the reduced instance can be used to efficiently compute a feasible solution for the original instance without changing the gap between the feasible solution and the optimum. Since every connected component of $G-(S\cup\mathsf{Mark}_1)$ is a $(\kappa/\varepsilon+1)\cdot(\rho+\eta)$-protrusion following the application of Rule 3, this gives a bound of $f(1/\varepsilon,\rho,\eta)$ (for some function f) on the size of each component of $G'-(S\cup\mathsf{Mark}_1)$ where G' is the graph obtained after exhaustively applying Rule 4.

3.2 Formal description of the algorithm

We now formally describe our kernelization algorithm. The algorithm begins by computing an approximate solution of size at most $\mathcal{O}(k)$, using the following lemma.

▶ **Lemma 22.** *There is an algorithm that, given a graph G and non-negative integers k,η,ρ, runs in time bounded by $\widetilde{h}(\eta+\rho)\cdot n^{\mathcal{O}(1)}$ for a computable function $\widetilde{h}$ and either correctly concludes that (G,k) is a no-instance of* Deletion (η/ρ)-Elimination*, or outputs a (ρ,η)-modulator of G of size at most $\widehat{c}\cdot\log(\rho+\eta)\cdot k$ for some constant $\widehat{c}$.*

If Lemma 22 returns that (G,k) is a no-instance of Deletion (η/ρ)-Elimination, then we return a trivial instance with $\mathcal{O}(k+\eta+\rho)$ vertices and an optimal solution value of at least $k+1$, as our output. Hereafter we assume that the output of Lemma 22 is a (ρ,η)-modulator S for G, of size at most $\widehat{c}\cdot\log(\rho+\eta)\cdot k$. We also assume that $|V(G-S)|\geq 2$, as otherwise, we can return G itself as a kernel with at most $\widehat{c}\cdot\log(\rho+\eta)\cdot k+2$ vertices. Using Observation 11, we compute an $(\eta,\mathcal{T}_\rho)$-decomposition, $\widetilde{\mathbb{D}}=(\widetilde{X},\widetilde{Y},\widetilde{T},\widetilde{f},\widetilde{g})$, of $G-S$ in time $\widehat{\ell}(\rho+\eta)n^{\mathcal{O}(1)}$ for some computable function $\widehat{\ell}$. Notice that using the above $(\eta,\mathcal{T}_\rho)$-decomposition and our assumption that $|V(G-S)|\geq 2$, we can compute an $(\eta',\mathcal{T}_\rho)$-decomposition (Observation 10), $\mathbb{D}=(X,Y,T,f,g)$, of $G-S$, where the following properties are satisfied:

- T is a rooted tree with at least two vertices,
- $f: X\to V(T)$ is a bijective function,
- $g:\mathcal{C}(G[Y])\to\mathsf{Lf}(T)$ and
- $\eta'=\mathsf{depth}(T)\leq\eta+2$.

We let r denote the root of the tree T. For $a \in V(T)$, we let $X_a = \{f^{-1}(a') \mid a' \in V(\mathsf{Pth}_T(a,r))\}$. Furthermore, we let G_a be the graph that appears "below" a, i.e., $V(G_a) = \{v \in X \mid f(v) \in \mathsf{desc}_T(a)\} \cup \{v \in Y \mid v \in V(C),$ where $C \in \mathcal{C}(G[Y])$ and $g(C) \in \mathsf{desc}_T(a)\}$, and $G_a = G[V(G_a)]$.

In the following we formalize the "monotonicity" of the problem in a way that will be useful in our proofs.

▶ **Observation 23.** *For every $\rho, \eta \in \mathbb{N}$, graph H, (ρ,η)-modulator $\widehat{S} \subseteq V(H)$ for H and every (not necessarily induced) subgraph $\widehat{H}$ of H, $\widehat{S} \cap V(\widehat{H})$ is a (ρ,η)-modulator for $\widehat{H}$.*

3.2.1 Identifying highly connected pairs

Let $\boxed{\phi_1 = k + \eta + 6\rho + 1}$. The first reduction rule adds an edge between a pair of (distinct) non-adjacent vertices $u, v \in V(G)$, where $\mathsf{sep}_G(u,v) \geq \phi_1 + 1$. Recall that $\mathsf{sep}_G(u,v)$ denotes the maximum number of internally vertex-disjoint u-v paths in G.

▶ **Reduction Rule 1.** *Let the input be (G,k). If there are distinct non-adjacent vertices $u, v \in V(G)$ such that $\mathsf{sep}_G(u,v) \geq \phi_1 + 1$, then return $(G + \{u,v\}, k)$.*

Notice that this rule can be applied exhaustively in polynomial time. In the next lemma, we analyze the effect of this reduction rule on the input instance.

▶ **Lemma 24.** *Let $u, v \in V(G)$ be non-adjacent vertices such that $\mathsf{sep}_G(u,v) \geq \phi_1 + 1$, and let $G' = G + \{(u,v)\}$. Then, the following hold:*
1. *Every (ρ,η)-modulator for G' is also a (ρ,η)-modulator for G.*
2. *Every (ρ,η)-modulator for G of size at most k is also a (ρ,η)-modulator for G'.*

The main consequence of Lemma 24 is that we can exhaustively apply Reduction Rule 1 without affecting solutions of size at most k for our instance.

In the context of our approximate kernel result, this is formalized below.

▶ **Lemma 25.** *Reduction Rule 1 is 1-safe.*

3.2.2 Bounding the degree of T

In what follows, we assume that Reduction Rule 1 is not applicable and continue to work with the (ρ,η)-modulator S and the $(\eta', \mathcal{T}_\rho)$-decomposition $\mathbb{D} = (X, Y, T, f, g)$, of $G - S$ where r denotes the root of the tree T. For each $C \in \mathcal{C}(G[Y])$, in time $2^{\mathcal{O}(\rho)}n$ we compute a tree decomposition of C, denoted by $\mathbb{T}_C = (T_C, \beta_C)$ of width at most 3ρ, where T_C is a rooted tree, using Proposition 6. To formalize our next marking scheme we recall the notion of "upward closure" for a vertex in $G - S$, which has also been used in previous related work [12, 1].

▶ **Definition 26.** Consider a vertex $v \in V(G - S)$. The *upward closure* of v in $G - S$ is the set $\mathsf{Up\text{-}Cls}(v)$, defined as follows: If $v \in X$, then $\mathsf{Up\text{-}Cls}(v) = X_{f(v)}$ and if $v \in Y$, then $\mathsf{Up\text{-}Cls}(v) = X_{f(g(C))} \cup \beta(b_v^*)$, where b_v^* is an (arbitrarily selected) node from T_C, such that $v \in \beta_C(b_v^*)$ and $\mathbb{T}_C = (T_C, \beta_C)$ is the tree decomposition computed above for the component $C \in \mathcal{C}(G[Y])$ that contains v.

We are now ready to present our first marking scheme.

▶ **Marking Scheme 1.** *Initialize $\mathsf{Mark}_1 = \emptyset$, and perform the following two steps (in the given order).*
1. *For every distinct $u, v \in S$, such that $\{u,v\} \notin E(G)$, compute a minimum $\{u\} - \{v\}$ separator $Z_{u,v}$ in G (in polynomial time), and for each $w \in Z_{u,v}$, add $\mathsf{Up\text{-}Cls}(w)$ to Mark_1.*

2. *For $C \in \mathcal{C}(G[Y])$, i) if $\mathsf{Mark}_1 \cap V(C) \neq \emptyset$, then let $M^*_C = \emptyset$, and ii) otherwise, let $\widehat{M}_C = \{b^*_v \in V(T_C) \mid v \in \mathsf{Mark}_1 \cap V(C) \neq \emptyset\}$, and $M^*_C = \mathsf{LCA\text{-}closure}_{T_C}(\widehat{M}_C)$. For each $C \in \mathcal{C}(G[Y])$ and $b \in M^*_C$, add $\beta_C(b)$ to Mark_1.*

As Reduction Rule 1 is not applicable, we obtain the following bound on $|\mathsf{Mark}_1|$.

▶ **Observation 27.** *We have $|\mathsf{Mark}_1| \leq 2 \cdot \binom{|S|}{2} \cdot \phi_1 \cdot (\eta' + 1) \cdot (\rho + 1)$.*

We define a few notations and give some results that will be useful in designing our next reduction rule. For a graph H and vertex set Q in $V(H)$, we let $\mathsf{elim\text{-}dist}_Q(H)$ the smallest integer λ for which H admits a $(\lambda, \mathcal{T}_\rho)$ decomposition, $\widetilde{\mathbb{D}} = (\widehat{X}, \widehat{Y}, \widehat{F}, \widehat{f}, \widehat{g})$, such that $Q \subseteq \widehat{Y}$, that is, Q appears in the *exterior* of this decomposition. If $Q = \emptyset$, then $\mathsf{elim\text{-}dist}_Q(H)$ is simply the elimination distance of H to $\mathcal{T}_\rho$. Notice that due to Observation 12, we can compute $\mathsf{elim\text{-}dist}_Q(H)$ in time bounded by $\widehat{h}(\eta, \rho, \mathsf{tw}(H)) \cdot |V(H)|^{\mathcal{O}(1)}$. In all our instantiations of H, we will have $\mathsf{tw}(H) = \mathcal{O}(\eta + \rho)$, and hence, the time required to compute $\mathsf{elim\text{-}dist}_Q(H)$ in these cases will be bounded by $\widehat{h}(\eta, \rho) \cdot n^{\mathcal{O}(1)}$

For every set $Z \subseteq V(G) \setminus S$, we define $\widehat{\mathcal{D}}_Z$ to be the set of connected components from $\mathcal{C}(G - (S \cup Z))$ that do not contain a vertex from Mark_1, i.e., $\widehat{\mathcal{D}}_Z = \{D \in \mathcal{C}(G - (S \cup Z)) \mid V(D) \cap \mathsf{Mark}_1 = \emptyset\}$. For such a Z and S, we now define a table $\mathcal{D}$ indexed by tuples comprising an element $s \in S \cup \{\iota\}$, a subset of S of size at most $\rho + 1$, a pair of disjoint subsets of Z plus a number.

▶ **Definition 28.** Fix $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$. For every $s \in S \cup \{\iota\}$, $S' \subseteq S \setminus \{s\}$ of size at most $\rho + 1$, disjoint subsets B_1, B_2 of Z, and $\widehat{\eta} \in [\eta]_0 \cup \{\infty\}$, $\mathcal{D}[s, S', B_1, B_2, \widehat{\eta}]$ denotes the set of all connected components $D \in \mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$ for which the following hold:
1. if $s \neq \iota$, then $s \in N(V(D))$,
2. $S' \cup B_1 \cup B_2 \subseteq N(V(D))$, and
3. if $\widehat{\eta} \neq \infty$ then $\mathsf{elim\text{-}dist}_{S' \cup B_2}(D) = \widehat{\eta}$, and otherwise, $\mathsf{elim\text{-}dist}_{S' \cup B_2}(D) > \widehat{\eta}$.

The following regarding the elements of $\widehat{\mathcal{D}}_Z$ will be handy.

▶ **Observation 29.** *Consider a set $Z \subseteq V(G) \setminus S$. For each $D \in \widehat{\mathcal{D}}_Z$, the following properties hold:*
1. *$N(V(D)) \cap S$ is a clique and*
2. *$\mathsf{elim\text{-}dist}_\emptyset(D) \leq \eta$.*

We will now describe a marking scheme, where, for a given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_B$, will create a subset of $\mathcal{D}$. We remark that for the actual execution of this marking scheme, we select Z and $\mathcal{D}$ later.

▶ **Marking Scheme 2.** *Given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$ as input, create a set $\mathsf{Mark}_2[Z, \mathcal{D}]$ as follows:*
1. *Initialize $\mathsf{Mark}_2[Z, \mathcal{D}] = \emptyset$.*
2. *For each $s \in S \cup \{\iota\}$, $S' \subseteq S \setminus \{s\}$ of size at most $\rho + 1$, disjoint subsets B_1, B_2 of Z, and $\widehat{\eta} \in [\eta]_0 \cup \{\infty\}$, add $\min\{|\mathcal{D}[s, S', B_{\mathsf{del}}, B_{\mathsf{int}}, B_{\mathsf{ext}}, \widehat{\eta}]|, \phi_1\}$-many elements from $\mathcal{D}[s, S', B_1, B_2, \widehat{\eta}]$ to $\mathsf{Mark}_2[Z, \mathcal{D}]$.*

Using the above marking scheme, we design our next reduction rule.

▶ **Reduction Rule 2.** *Given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$, if $\mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$ is non-empty, then select $D \in \mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$ and return $(G - V(D), k)$.*

We say that for a given Z and $\mathcal{D}$, Reduction Rule 2 is *applicable* if $\mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$ is non-empty. Notice that given $Z \subseteq V(G) \setminus S$ and a subset $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$, $|\mathsf{Mark}_2[Z, \mathcal{D}]|$ can be bounded by $(|S|+1)^{\rho+3} \cdot 3^{|Z|} \cdot (\eta+3) \cdot \phi_1$. In all our applications of Reduction Rule 2, we will have that $|Z| \leq \eta' + 1 = \eta + 3$, which implies an upper bound of $\boxed{\phi_2 = (|S|+1)^{\rho+3} \cdot 3^{\eta+3} \cdot (\eta+3) \cdot \phi_1}$ on the size of $|\mathsf{Mark}_2[Z, \mathcal{D}]|$ in all our instantiations of Z. Moreover, it is straightforward to see that for all such Z, $\mathsf{Mark}_2[Z, \mathcal{D}]$ can be computed in time $\mathcal{O}(\phi_2 \cdot n^{\mathcal{O}(1)})$.

The exhaustive application of Reduction Rule 2 essentially says that any element in $\mathcal{D}$ that is "untouched" by Marking Scheme 2 must be deleted. The intuition is that for any component that is deleted by this rule, sufficiently many "similar representatives" are preserved, thus also preserving solutions of size at most k. This is formalized in the following lemma.

▶ **Lemma 30.** *Given $Z \subseteq V(G) \setminus S$ and $\mathcal{D} \subseteq \widehat{\mathcal{D}}_Z$, for which there is a $D \in \mathcal{D} \setminus \mathsf{Mark}_2[Z, \mathcal{D}]$, let $G' = G - V(D)$ be the graph obtained by an application of Reduction Rule 2. Then, a set $S^* \subseteq V(G)$ of size at most k is a (ρ, η)-modulator for G if and only if $S^* \cap V(G')$ is a (ρ, η)-modulator for G'.*

We now argue that Reduction Rule 2 can be used in our approximate kernel.

▶ **Lemma 31.** *Reduction Rule 2 is 1-safe.*

Recall the $(\eta', \mathcal{T}_\rho)$-decomposition $\mathbb{D} = (X, Y, T, f, g)$ of $G - S$ that we currently have. Using Reduction Rule 2, we are able to reduce $|V(T)|$, $|\mathcal{C}(G[Y])|$, and for each $C \in \mathcal{C}(G[Y])$, bound $|\mathcal{C}(C - \mathsf{Mark}_1)|$ by appropriately choosing (several values for) Z and $\mathcal{D}$. This is argued as follows.

Reducing $|V(T)|$ and $|\mathcal{C}(G[Y])|$

Recall that $\mathsf{depth}(T) \leq \eta' = \eta + 2$. Thus, the number of vertices in T is at most $\eta + 2$ times the maximum degree of a vertex in T. We will next focus on bounding degree of vertices in T, in particular, we will bound the maximum number of children for a vertex in T. Consider a vertex $a \in V(T)$. Recall that $X_a = \{f^{-1}(a') \mid a' \in V(\mathsf{Pth}_T(a, r))\}$, where r is the root of T.

We apply Reduction Rule 2, by choosing $Z = X_a$ with the subset $\mathcal{D} = \mathcal{D}_a \subseteq \widehat{\mathcal{D}}_{X_a}$ defined as follows:

- If a is a non-leaf vertex in T, then we set $\mathcal{D}_a = \{D \in \mathcal{C}(G_{a'}) \mid V(D) \cap \mathsf{Mark}_1 = \emptyset, a' \in V(T) \text{ and } a = \mathsf{par}_T(a')\}$.
- Otherwise, a is a leaf of T and we set $\mathcal{D}_a = \{C \in \mathcal{C}(G[Y]) \mid V(D) \cap \mathsf{Mark}_1 = \emptyset \ , \ g(C) = a\}$.

Intuitively speaking, we have defined $\mathcal{D}_a$ as the subset of $\widehat{\mathcal{D}}_{X_a}$ that comprises of those connected components that appear "below" the children of a in T. Moreover, when a is a leaf, $\mathcal{D}_a$ will be those connected components $C \in G[Y]$, for which $g(C) = a$.[4]

Note that $\mathcal{D}_a \subseteq \widehat{\mathcal{D}}_{X_a}$ (see Definition 28 to recall the definition of $\widehat{\mathcal{D}}_Z$ for our choice of $Z = X_a$). Also, notice that the number of children of a in T is bounded by $|\mathcal{D}_a| + |\mathsf{Mark}_1|$. Recall that for $a \in V(a)$, $|X_a| \leq \eta$. This together with Lemma 22 and Lemma 30 implies the following result.

[4] Recall our assumption that for each $C \in G[Y]$, $g(C) \in \mathsf{Lf}(T)$.

▶ **Observation 32.** *For each $a \in V(T)$, in time bounded by $(k \cdot \log(\eta + \rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)} \cdot n^{\mathcal{O}(1)}$ we can either apply Reduction Rule 2 for $Z = X_a$ and $\mathcal{D} = \mathcal{D}_a$, or correctly conclude that it is not applicable. Moreover, if for no $a \in V(T)$, the reduction rule is applicable, then $|V(T)| \leq \eta' \cdot (\phi_2 + |\mathsf{Mark}_1|) = (k \cdot \log(\eta+\rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)}$ and $|\mathcal{C}(G[Y])| \leq |V(T)| \cdot (\phi_2 + |\mathsf{Mark}_1|) = (k \cdot \log(\eta + \rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)}$.*

Note that to obtain our kernel, it is enough to bound the number of vertices appearing in the connected components in $G[Y]$ by our stated bound. To this end, we will first bound the number of connected components in $G[Y] - \mathsf{Mark}_1$ by this bound. Due to Observation 32, we have already bounded the number of connected components in $\mathcal{C}(G[Y])$ and therefore, to achieve the above, it is enough to bound, for each $C \in \mathcal{C}(G[Y])$, the number of connected components in $C - \mathsf{Mark}_1$.

Bounding $|\mathcal{C}(C - \mathsf{Mark}_1)|$, for $C \in G[Y]$

Consider a connected component $C \in G[Y]$. Recall that we have already computed a tree decomposition $\mathbb{T}_C$ of width at most $6 \cdot \rho$ for C. If $V(C) \cap \mathsf{Mark}_1 = \emptyset$, then $|\mathcal{C}(C - \mathsf{Mark}_1)| = 1$. Thus we hereafter assume that $V(C) \cap \mathsf{Mark}_1 \neq \emptyset$. The above implies that $M^*_C \neq \emptyset$ (see Marking Scheme 1). By the construction of Mark_1, notice that $\mathsf{Mark}_1 \cap V(C) = \cup_{b \in M^*_C} \beta_C(b)$. Let $\mathcal{H}_C$ be the set of connected components in $T_C - M^*_C$, i.e., $\mathcal{H}_C = \mathcal{C}(T_C - M^*_C)$. For each $H \in \mathcal{H}_C$, let $\widetilde{\mathcal{D}}_H$ be the set of connected components of $G[V_H]$, where $V_H = (\cup_{b \in V(H)} \beta_C(b)) \setminus \mathsf{Mark}_1$. For $H \in \mathcal{H}_C$, (intuitively speaking) we let $Z[H]$ be set of vertices that appear in the bags that neighbor H in T_C and the vertices from X, that appear in the root-to-$g(C)$ path in F (recall that $g(C) \in \mathsf{Lf}(F)$). Formally, for $H \in \mathcal{H}_C$, let $Z[H] = (\cup_{b \in N_T(V(H))} \beta_C(b)) \cup X_{g(C)}$. We next summarize some properties of $\mathcal{H}_C$ and the sets we constructed above, which follows from their definitions and Observation 2.

▶ **Observation 33.** *We have $|\mathcal{H}_C| \leq 2|\mathsf{Mark}_1|$, and for each $H \in \mathcal{H}_C$, the following holds:*
1. $|N_T(V(H))| \leq 2$,
2. $Z[H] \subseteq \mathsf{Mark}_1 \cup X \subseteq V(G) \setminus S$, *where* $|Z[H]| \leq 2 \cdot (6\rho + 1) + \eta' = 2 \cdot (6\rho + 1) + \eta + 2$,
3. *for each* $D \in \widetilde{\mathcal{D}}_H$, $N(V(D)) \subseteq S \cup Z[H]$,
4. $\widetilde{\mathcal{D}}_H \subseteq \widehat{\mathcal{D}}_{Z[H]}$, *and*
5. *for each* $D \in \mathcal{C}(C - \mathsf{Mark}_1)$, *there is some* $H \in \mathcal{H}_C$, *such that* $D \in \widetilde{\mathcal{D}}_C$.

From the above observation, to bound the size of $\mathcal{C}(C - \mathsf{Mark}_1)$, it is enough to bound $|\widetilde{\mathcal{D}}_H|$, for each $H \in \mathcal{H}_C$. For each $H \in \mathcal{H}_C$, we use Reduction Rule 2 (if applicable), for $Z = Z[H]$ and $\mathcal{D} = \widetilde{\mathcal{D}}_H$. The above, together with Lemma 30 implies the following.

▶ **Observation 34.** *For each $C \in \mathcal{C}(G[Y])$ and $H \in \mathcal{H}_C$, in time bounded by $(k \cdot \log(\eta + \rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)} \cdot n^{\mathcal{O}(1)}$, we can either apply Reduction Rule 2 for $Z = Z[H]$ and $\mathcal{D} = \widetilde{\mathcal{D}}_H$, or correctly conclude that it cannot be applied. Moreover, if for no $C \in \mathcal{C}(G[Y])$ and $H \in \mathcal{H}_C$, the reduction rule is applicable, $|\mathcal{C}(G[Y] - \mathsf{Mark}_1)|$ can be bounded by $(k \cdot \log(\eta + \rho))^{\mathcal{O}(\rho)} \cdot 2^{\mathcal{O}(\eta)}$.*

That is, if we have exhaustively applied the first two reduction rules, then we will have bounded $|X|$ and the number of connected components in $G[Y] - \mathsf{Mark}_1$ by $f(\eta, \rho) \cdot k^{\mathcal{O}(\rho)}$ for some comptuable function f. In the rest of the paper, our goal is to bound the size of the components in $G[Y] - \mathsf{Mark}_1$.

3.2.3 Converting components in $G - (S \cup \mathsf{Mark}_1)$ to protrusions

We begin by arguing that for any component C of $G - (S \cup \mathsf{Mark}_1)$, any (ρ, η)-modulator contains "almost all" neighbors of C inside S.

▶ **Lemma 35.** *For every $C \in (G - (S \cup \mathsf{Mark}_1))$, any (ρ, η)-modulator for G must contain at least $|N(C) \cap S| - \eta - \rho - 2|$ vertices of $N(C) \cap S$.*

We have already noted that every component in $(G - (S \cup \mathsf{Mark}_1)$ has at most $2\rho + \eta' + 1 = 2\rho + \eta + 3$ neighbors in Mark_1 (see discussion following Observation 27). This fact, plus Lemma 35 motivate the following reduction rule.

▶ **Reduction Rule 3.** *If there is a $C \in (G - (S \cup \mathsf{Mark}_1))$ such that $|N(C)| \geq (\lceil 1/\varepsilon \rceil + 1) \cdot (3\rho + 2\eta + 5)$, then return $(G - (N(C), k - (|N(C)| - 3\rho - 2\eta - 5))$. That is, delete $N(C)$ and reduce the parameter by $|N(C)| - 3\rho - 2\eta - 5|$.*

We note that if Reduction Rule 3 is not applicable, then every connected component in $(G - (S \cup \mathsf{Mark}_1)$ has a neighborhood of size at most $\xi = \mathcal{O}(1/\varepsilon(\rho + \eta))$. The intuition behind the this reduction rule is that if $N(C)$ is sufficiently large compared to $3\rho + 2\eta + 5$, then, any (ρ, η)-modulator for G contains "most" vertices of $N(C)$ and we can add all vertices of $N(C)$ to the solution while charging the few error vertices to the many "correct" vertices.

▶ **Lemma 36.** *Reduction Rule 3 is $(1 + \varepsilon)$-safe.*

3.2.4 Reducing protrusions in $G - (S \cup \mathsf{Mark}_1)$

At this point, we have that every component of $G - (S \cup \mathsf{Mark}_1)$ has treewidth at most $\eta + \rho + 1 \leq \xi$ and moreover, has neighborhood at most ξ. In other words, every component of $G - (S \cup \mathsf{Mark}_1)$ is a ξ-protrusion, where ξ is a constant depending only on $1/\varepsilon, \rho, \eta$. In what follows, we use tools from protrusion reduction and replacement to reduce the size of each component of $G - (S \cup \mathsf{Mark}_1)$. In particular, we use the *lossless protrusion replacer* of Fomin et al. [11].

▶ **Definition 37** ([11]; Lossless Protrusion Replacer)**.** A lossless protrusion replacer for a min-CMSO vertex subset problem Π is a family of algorithms, with one algorithm for every constant r. The r'th algorithm has the following specifications. There exists a constant r' (which depends on r) such that given an instance G and an r-protrusion X in G of size at least r', the algorithm runs in time $\mathcal{O}(|X|)$ and outputs an instance G' with the following properties:

1. G' is obtained from G by replacing X by a r-boundaried graph X' with fewer than r' vertices and thus $|V(G')| < |V(G)|$.
2. $\mathsf{OPT}(G') \leq \mathsf{OPT}(G)$.
3. There is an algorithm that runs in $\mathcal{O}(|X|)$ time and given a feasible solution S' to G' outputs a set $X^* \subseteq X$ such that $S = (S' \setminus X') \cup X^*$ is a feasible solution to G and $|S| \leq |S'| + \mathsf{OPT}(G) - \mathsf{OPT}(G')$.

Since graphs admitting an $(\eta, \mathcal{T}_\rho)$-decomposition are minor-closed, CMSO-expressible and exclude at least one planar graph as a minor (e.g., an $(\eta + \rho + 2) \times (\eta + \rho + 2)$ grid), it follows from [11] that the Deletion (η/ρ)-Elimination problem has a lossless protrusion replacer. Moreover, using reasoning similar to that in Lemmas 25, 31 and 36 (with the solution-lifting algorithm given by the third property listed in Definition 37), we conclude that this reduction is 1-safe. Hence, we infer that by exhaustively invoking this subroutine with $r = \chi$, we are able to reduce the size of every connected component in $G - (S \cup \mathsf{Mark}_1)$ to some r' which is a function of r, which in turn is only a function of $1/\varepsilon, \rho$ and η. This gives us a bound of $f(\eta, \rho, 1/\varepsilon) \cdot k^{g(\rho)}$ on the size of the final reduced instance (for some functions f and g), proving Theorem 1.

4 Conclusion and future work

The central follow-up question to our work specifically on Deletion (η/ρ)-Elimination is whether the approximation in our result can be avoided and instead a "standard" polynomial kernel of size $f(\eta,\rho)k^{g(\rho)}$ can be obtained for the problem.

On a broader level, our work motivates the interpolation between other results centered around treedepth and treewidth in parameterized complexity. For example, consider a graph problem that has an $f(td)\cdot n^{\mathcal{O}(1)}$ algorithm parameterized by the treedepth td for some function f, but has a best-possible (or best-known) algorithm with running time $n^{g(tw)}$ for some function h when parameterizing by the treewidth tw (e.g., Grundy Coloring [2]). Our work naturally motivates the study of these problems with the aim of obtaining algorithms with running time $f(\eta)\cdot n^{g(\rho)}$ on graphs that have elimination distance η to the class $\mathcal{T}_\rho$. Such algorithms would provide interpolations between known results that are based on these well-studied width measures.

References

1. Akanksha Agrawal and M. S. Ramanujan. On the parameterized complexity of clique elimination distance. 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 1:1–1:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.1.
2. Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. 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 14:1–14:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.14.
3. Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016.
4. Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017. doi:10.1007/s00453-016-0235-7.
5. Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
6. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
7. Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.
8. Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.
9. Samuel Fiorini, Gwenaël Joret, and Ugo Pietropaoli. Hitting diamonds and growing cacti. In *Integer Programming and Combinatorial Optimization, 14th International Conference, IPCO 2010, Lausanne, Switzerland, June 9-11, 2010. Proceedings*, pages 191–204, 2010. doi:10.1007/978-3-642-13036-6_15.
10. Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM J. Discrete Math.*, 30(1):383–410, 2016. doi:10.1137/140997889.
11. Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-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, 2012. doi:10.1109/FOCS.2012.62.

12 Archontia C. Giannopoulou, P. Jansen Bart M. Daniel Lokshtanov, and Saket Saurabh. Uniform kernelization complexity of hitting forbidden minors. *ACM Trans. Algorithms*, 13(3):35:1–35:35, 2017. doi:10.1145/3029051.

13 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. 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 1757–1769. ACM, 2021. doi:10.1145/3406325.3451068.

14 Gwenaël Joret, Christophe Paul, Ignasi Sau, Saket Saurabh, and Stéphan Thomassé. Hitting and harvesting pumpkins. *SIAM J. Discrete Math.*, 28(3):1363–1390, 2014. doi:10.1137/120883736.

15 Tuukka Korhonen. Single-exponential time 2-approximation algorithm for treewidth. In *FOCS*, 2021.

16 Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014.

17 Alexander Lindermayr, Sebastian Siebertz, and Alexandre Vigny. Elimination distance to bounded degree on planar graphs. 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 65:1–65:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.65.

18 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization – Preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161. Springer, 2012.

19 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. *CoRR*, abs/1604.04111, 2016. arXiv:1604.04111.

20 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 224–237, 2017. doi:10.1145/3055399.3055456.

21 Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

22 Geevarghese Philip, Venkatesh Raman, and Yngve Villanger. A quartic kernel for pathwidth-one vertex deletion. In *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010 Revised Papers*, pages 196–207, 2010. doi:10.1007/978-3-642-16926-7_19.

Hardness of Learning Boolean Functions from Label Proportions

Venkatesan Guruswami
Department of EECS and Simons Institute for the Theory of Computing,
University of California, Berkeley, CA, USA

Rishi Saket
Google Research India, Banglaore, India

Abstract

In recent years the framework of learning from label proportions (LLP) has been gaining importance in machine learning. In this setting, the training examples are aggregated into subsets or *bags* and only the average label per bag is available for learning an example-level predictor. This generalizes traditional PAC learning which is the special case of unit-sized bags. The computational learning aspects of LLP were studied in recent works [21, 22] which showed algorithms and hardness for learning halfspaces in the LLP setting. In this work we focus on the intractability of LLP learning Boolean functions. Our first result shows that given a collection of bags of size at most 2 which are consistent with an OR function, it is NP-hard to find a CNF of constantly many clauses which *satisfies* any constant-fraction of the bags. This is in contrast with the work of [21] which gave a (2/5)-approximation for learning ORs using a halfspace. Thus, our result provides a separation between constant clause CNFs and halfspaces as hypotheses for LLP learning ORs.

Next, we prove the hardness of satisfying more than $1/2 + o(1)$ fraction of such bags using a t-DNF (i.e. DNF where each term has $\leq t$ literals) for any constant t. In usual PAC learning such a hardness was known [15] only for learning noisy ORs. We also study the learnability of parities and show that it is NP-hard to satisfy more than $(q/2^{q-1} + o(1))$-fraction of q-sized bags which are consistent with a parity using a parity, while a random parity based algorithm achieves a $(1/2^{q-2})$-approximation.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Learning from label proportions, Computational learning, Hardness, Boolean functions

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.37

Funding *Venkatesan Guruswami*: Research supported in part by a Simons Investigator award and NSF grants CCF-2228287 and CCF-2211972.

1 Introduction

In common machine learning applications, one is required to train a classifier using some training set of (vectors, label)-pairs to predict the label of vectors sampled from the same (or a similar) distribution as the training set. A typical approach is to optimize the classifier to predict correctly on the training set to ensure that the classifier has good predictive performance over the target distribution. This optimization view is captured by the *probably approximately correct* (PAC) learning framework [23].

In setting of *learning from label proportions* (LLP), the training set consists of subsets or *bags* of vectors along with the sum or average of the labels of vectors in each bag. The goal is to train a model to predict the labels for vectors. As before, one would want the model to firstly predict as correctly as possible on the training bags. One measure of such performance is the fraction of *satisfied* bags i.e., those on which the predicted average label matches the given average label i.e., the label proportion. Note that traditional PAC learning is the special case of LLP with only unit-sized bags.

© Venkatesan Guruswami and Rishi Saket;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 37; pp. 37:1–37:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

LLP is motivated by applications in which only the aggregated labels for bags of vectors are available. This may be to preserve the privacy [20, 24, 16] of labels, due to lack of instrumentation to obtain labels [9] or high labeling costs [8]. Other examples of LLP applications have been in medical image classification [14, 5, 17] where small bag sizes – in the range of 10 to 50 – are typically more relevant (see Sec 1.2 of [4]).

The work of [21] studied LLP from the computational learning perspective on bags of size ≤ 2. The LLP learning goal is the following: given a collection of bags consistent with some function from a target concept class, compute a hypothesis satisfying the most number of bags. With this objective, [21] showed a $(1/2 + o(1))$-factor hardness for LLP learning a halfspace using any function of constantly many halfspaces on bags of size at most 2. From the algorithmic side on such bags [21] gave a $(2/5)$-factor approximation for LLP learning a halfspace using halfspace, based on rounding a semi-definite programming (SDP) relaxation. Subsequently, [22] proved a strengthened $(4/9 + o(1))$-factor hardness for LLP learning a halfspace using any function of constantly many halfspaces on bags of size at most 2, a corresponding $(1/q + o(1))$-factor hardness for bags of size at most any constant $q \in \mathbb{Z}^+$, and extended the algorithmic result of [21] showing a $(1/12)$-approximation on bags of size at most 3.

Since halfspaces capture OR formulas, the algorithmic results of [21, 22] apply to learning OR formulas using halfspaces. Moreover, the $(1/2 + o(1))$-factor hardness of [21] on bags of size ≤ 2 also holds for LLP learning an OR using any function of constantly many halfspaces. Typically however, one would like to learn an OR using an OR or similar Boolean functions such as ℓ-clause CNF formulas (OR is 1-clause CNF), rather than halfspaces. This raises the following question

Can we achieve constant-factor algorithmic approximations for LLP learning OR *using* OR *or constant-clause* CNF*?*

In our first result, we answer the above question in the negative.

▶ **Theorem 1.** *For any constants $\delta > 0, \ell \in \mathbb{Z}^+$, given a collection of bags which are of size at most 2, and whose label proportions are consistent with some* OR*, it is NP-hard to compute an ℓ-clause* CNF *that satisfies δ-fraction of the bags.*

The above theorem is proved in Sec. 3. We find the result interesting since it (along with the algorithmic results of [21, 22]) proves a separation between constant clause CNFs – in particular ORs – and halfspaces as hypotheses for learning ORs.

We also study the LLP learnability of OR using as hypothesis ℓ-DNF formulas i.e., DNF where each term is a conjunction of at most ℓ literals. While OR is 1-DNF, for $\ell \geq 2$, ℓ-DNFs are not contained in halfspaces, therefore have the possibility of yielding better approximations. However, our second result below (proved in Sec. 4) essentially rules out this possibility.

▶ **Theorem 2.** *For any constants $\delta > 0, \ell \in \mathbb{Z}^+$, given a collection of bags which are of size at most 2, and whose label proportions are consistent with some* OR*, it is NP-hard to compute an ℓ-*DNF *that satisfies $(1/2 + \delta)$-fraction of the bags.*

Note that while the hardness factor achieved above is weaker than that of Theorem 1, no inapproximability is known for the real analogue of Theorem 2 i.e., LLP learning halfspaces using *polynomial thresholds*.

While the works of [21, 22] studied the LLP learnability of halfspaces, the corresponding problem over finite fields has not been studied. In particular, the $\mathbb{F}_2$-version of this problem is equivalent to the LLP learnability of parities using parities over the Boolean domain.

Parities are a fundamental class of Boolean functions which makes this problem of significant interest as well. Our next result however, shows that this is hard to approximate, with the inapproximability growing exponentially as the bag size increases.

▶ **Theorem 3.** *For any constants $\delta > 0, q \in \mathbb{Z}^+$ $(q \geq 2)$, given a collection of bags which are of size at most q, and whose label proportions are consistent with some parity, it is NP-hard to compute a parity that satisfies $(q/2^{q-1} + \delta)$-fraction of the bags.*

The above is proved in Section 5. The hardness factor is asymptotically close (for large q) to the $(1/2^{q-2})$-approximation for this problem using random parities described in Sec. 6.

1.1 Previous Related Work

The formalization of the LLP framework was first done in the work of [25] who proved generalization error bounds for classifiers for any distribution over (bag, label-proportion)-pairs, though their bag-level objective was a relaxed notion – useful for studying LLP with large bag sizes – of the strict bag satisfaction used in [21, 22] and our work. Related recent works [6, 7] have shown bag-to-instance classification generalization error bounds. The study of LLP learnability of specific function classes has nevertheless been fairly sparse, apart from the works of [21, 22] whose contributions have been described earlier in this section.

The learnability of small Boolean formulas has been extensively studied in traditional PAC learning. It is well known that an OR can be efficiently learnt by an OR up to arbitrary accuracy. On the other hand, [15] proved a $(1/2+o(1))$-factor hardness for learning a 2-clause CNF using constant clause CNF, and the same hardness factor for learning a noisy OR using ℓ-DNF for any constant ℓ. The work of [10] proved the same hardness for learning noisy OR with a halfspace as hypothesis. These results were further generalized by [11] who proved the same hardness factors for learning 2-clause CNF and noisy OR using any function of constantly many halfspaces as hypothesis. Similar to OR, parities can also be efficiently learnt by parities using Gaussian elimination over $\mathbb{F}_2$. On the other hand, the $(1/2 + o(1))$-factor hardness for noisy Max-3-Lin by [13] implies the same hardness factor for learning a noisy parity using a parity. Note that all the $(1/2+o(1))$-factor hardness results are tight since one of the constant 0 or 1 functions trivially obtain $(1/2)$-approximation for learning Boolean valued functions. However, this trivial threshold does not hold in the LLP setting since the constant functions are not guaranteed to satisfy even one bag.

The above hardness results carry over to the LLP setting for the special case when all bags are unit-sized. However, we prove hardness of approximating the problems of LLP learning OR and parity *without* any noise, which are tractable in the usual PAC case, thereby showing a qualitative difference between the LLP and PAC settings.

1.2 Overview of Our Techniques

Proof of Theorem 1. Our reduction is from bipartite Label-Cover [1] with N and M as the sizes of the smaller and larger label sets respectively, and is similar to that of [15] for the hardness of learning noisy OR. The high-level approach is to have one coordinate for each vertex-label pair on the larger (right) side of the Label-Cover instance, i.e. the variables are $x_{v,i}$ for $v \in V$ and $i \in [M]$. Fix a random sample of $2t$ vertices $\{\hat{v}_1, \ldots, \hat{v}_t, \tilde{v}_1, \ldots, \tilde{v}_t\}$ from a neighborhood of a left vertex u. For simplicity assume that the projection constraints between u and each of the $2t$ vertices are the same i.e, for each label $j \in [N]$ for u there is a subset $S_j \subseteq [M]$ such that assigning any of the $2t$ vertices with a label from S_j satisfies that edge.

A 2-sized bag with label proportion 1/2 is sampled by letting $J \subseteq [N]$ be a random subset, and for the first point $\mathbf{x}$ setting only the coordinates $\{x_{\hat{v}_r,i} \mid \pi_{\hat{v}_r u}(i) \in J, r \in [t]\}$ to be 1, and for the second point $\mathbf{z}$ only the coordinates $\{z_{\tilde{v}_r,i} \mid \pi_{\tilde{v}_r u}(i) \in \overline{J}, r \in [t]\}$ to be 1. It is easy to see in the YES case that an OR of exactly the coordinates $(v, \rho(v))$ – where ρ is the satisfying labeling – for each right vertex v, satisfies all such bags. For the NO case, we illustrate the analysis of an OR formula $\mathcal{C}$ which satisfies some constant fraction of the bags. From the $o(1)$-Hamming weight of the points, one can assume that $\mathcal{C}$ has no negated literals. If $\mathcal{C}$ has no coordinates of the $2t$ vertices (empty case) then the bag is anyway not satisfied as $\mathcal{C}$ evaluates to 0 on both points.

On the other hand, if a sufficiently large number of these vertices have a corresponding variable in $\mathcal{C}$ (dense case), then elementary probabilistic arguments yield at least two vertices among the $2t$ which have their pre-decided distinguished variables in $\mathcal{C}$ with the same projection, leading to a good randomized labeling to the Label-Cover. A key idea for ensuring that this analysis goes through is to sample t u.a.r. from $\{1, \ldots, 2^T\}$ for some large T so that for each u and most values of t, nearly all samples of the $2t$ vertices are either the empty case or the dense case.

Proof of Theorem 2. The hardness reduction is similar to the above, except that we need to ensure that a significant fraction of vertices have at least one term in which all the positive literals correspond to its coordinates, while none of the negated literals do. Thereafter a similar analysis as the previous case goes through. However, for ensuring this property we introduce an additional distribution over bags of size 1 and label proportion 1, essentially saying that for each vertex the point with all its coordinates to 1 and the rest to 0 should be 1-labeled. Due to this we obtain a $(1/2 + o(1))$-factor hardness in this case.

Proof of Theorem 3. While the reductions above create points in a bag whose active coordinates span the edges of the label-cover, in the parity case we can add homogeneous $\mathbb{F}_2$-linear *folding* constraints which ensure consistency of labels across edges via a reduction from the non-bipartite *Smooth* Label-Cover [12]. It is sufficient to then describe a *dictatorship test* (see Chap. 7 of [18]) on the M coordinates of a single vertex. Our dictatorship test is a distribution over q-sized bags, i.e., q points in the hypercube $\mathbb{F}_2^M$, sampled as follows: independently for each $i \in M$, set exactly one of the q points to 1 in the i'th coordinate and the rest to 0. All these bags have target label proportion $1/q$ which is satisfied by any dictator, i.e., parity given by a single coordinate. On the other hand, any parity over a much larger number K of coordinates will induce a near-uniform distribution over the q-sized vector of labeling to the points of a random bag. In fact, this is close to the uniform distribution over the points of $\mathbb{F}_2^q$ with even or odd (depending only on q and K) number of non-zero coordinates. It is then easy to see that such a distribution will satisfy the $1/q$ label proportion of the bags with probability $\approx q/2^{q-1}$.

The algorithm for this problem first does Gaussian elimination for all the linear constraints given by bags of label proportion $\{0, 1\}$ and obeying the bag-level parity constraints for the remaining bags. It then chooses a random parity from the remaining coordinates. We show that this yields an $1/2^{q-2}$ approximation.

2 Preliminaries

2.1 Problem Definitions

Consider the space $\{0,1\}^d$ for some $d \in \mathbb{Z}^+$ and some function $f : \{0,1\}^d \to \{0,1\}$. For $B \subseteq \{0,1\}^d$, define $\sigma(B, f) := |\{\mathbf{x} \in B \mid f(\mathbf{x}) = 1\}| / |B|$ to be the corresponding label proportion.

An instance $\mathcal{I}$ of LLP-OR$[q]$ is given by a collection $\mathcal{B} := \{(B_j, \sigma_j)\}_{j=1}^m$ of bags and their label proportions where each bag is of size at most q. The goal is to find an OR function $h(\mathbf{x})$ which satisfies the most bags of $\mathcal{B}$ i.e., maximize $|\{j \in [m] \mid \sigma_j = \sigma(B_j, h)\}|$.

An instance $\mathcal{I}$ of LLP-PARITY$[q]$ is similar to the above except that the goal accordingly is to compute a parity maximizing the number of satisfied bags. Since the XOR is simply an addition over $\mathbb{F}_2$ we shall think of the Boolean values as elements of $\mathbb{F}_2$ in this case.

2.2 Label Cover

▶ **Definition 4.** *An instance of $\mathcal{L}$ of* Label-Cover *is given by $(G(U, V, E \subseteq V \times U), M, N, \{\pi_{vu} : [M] \to [N]\}_{e=(v,u)\in E})$ where $G(U, V, E)$ is a bi-regular bipartite graph. A labeling ρ assigning labels from $[N]$ to U and $[M]$ to V satisfies an edge (u, v) iff $\pi_{vu}(\rho(v)) = \rho(u)$. The goal is to find a labeling satisfying the most number of edges.*

The following well-known inapproximability of Label-Cover follows from the the PCP Theorem [3, 2] along with the Parallel Repetition Theorem [19].

▶ **Theorem 5.** *For any constant $\xi > 0$ there exist M and N such that it is NP-hard, given an* Label-Cover *instance $\mathcal{L}(U, V, E, M, N, \{\pi_{vu}\}_{e=(v,u)\in E})$ to distinguish between: (i)* YES case: *there is labeling that satisfies all edges in E, or (ii)* NO case: *Any labeling satisfies at most ξ fraction of the edges of E.*

2.3 Smooth Label Cover

Unlike the standard bipartite version in Definition 4 we also use a non-bipartite version with useful structural properties defined below.

▶ **Definition 6.** *An instance of* Smooth-Label-Cover *$\mathcal{L}(G(V, E), N, M, \{\pi_{ev} \mid e \in E, v \in e\})$ consists of a regular connected (undirected) graph $G(V, E)$ with vertex set V and edge set E. Every edge $e = (v_1, v_2)$ is associated with projection functions $\{\pi_{ev_i}\}_{i=1}^2$ where $\pi_{ev_i} : [M] \to [N]$. A vertex labeling is a mapping defined on $\rho : V \to [M]$. A labeling ρ satisfies edge $e = (v_1, v_2)$ if $\pi_{ev_1}(\rho(v_1)) = \pi_{ev_2}(\rho(v_2))$. The goal is to find a labeling which satisfies the maximum number of edges.*

The following theorem is proved in Appendix A of [12].

▶ **Theorem 7.** *There exists a constant $c_0 > 0$ such that for any constant integer parameters $Q, R \geq 1$, it is* NP*-hard to distinguish between the following two cases for a* Smooth Label Cover *instance $\mathcal{L}(G(V, E), N, M, \{\pi_{ev} \mid e \in E, v \in e\})$ with $M = 7^{(Q+1)R}$ and $N = 2^R 7^{QR}$:*

- (YES Case) *There is a labeling that satisfies every edge.*
- (NO Case) *Every labeling satisfies less than a fraction $2^{-c_0 R}$ of the edges.*

In addition, the instance $\mathcal{L}$ satisfies the following properties:

- (Smoothness) *For any vertex $w \in V$, $\forall i, j \in [M],\ i \neq j,\ \Pr_{e\sim w}[\pi_{ew}(i) = \pi_{ew}(j)] \leq 1/Q$, where the probability is over a randomly chosen edge incident on w.*
- (Weak Expansion) *For any $\delta > 0$, let $V' \subseteq V$ and $|V'| = \delta \cdot |V|$, then the number of edges among the vertices in $|V'|$ is at least $\delta^2|E|$.*

3 Hardness of LLP Learning OR using ℓ-clause CNF

We prove the following hardness reduction which along with Theorem 5 implies Theorem 1.

▶ **Theorem 8.** *For any constants $\delta > 0$ and $\ell \in \mathbb{Z}^+$, there is a polynomial time reduction from an instance $\mathcal{L}$ of* Label-Cover *to an* LLP-OR[2] *instance $\mathcal{B}$ s.t.*
YES Case: *If $\mathcal{L}$ is YES instance then there is an* OR *consistent with all the bags of $\mathcal{B}$.*
NO Case: *If $\mathcal{L}$ is a NO instance then there is no ℓ-clause* CNF *formula satisfying at least δ-fraction of the bags.*

We begin with the following useful technique. Let $T \geq 10$ be a large integer and consider the set $\mathcal{T} = \{2, 4, \ldots, 2^T\}$. For any $s \in \mathbb{R}^+$ define the subsets $L(s), R(s) \subseteq \mathcal{T}$ as

$$L(s) := \{t \in \mathcal{T} \mid t \leq s/\sqrt{T}\} \quad \text{and} \quad R(s) := \{t \in \mathcal{T} \mid t \geq s \cdot \sqrt{T}\}. \tag{1}$$

We have the following simple lemma:

▶ **Lemma 9.** *For any $s \in \mathbb{R}^+$, $L(s) \cap R(s) = \emptyset$ and $|L(s)| + |R(s)| \geq |\mathcal{T}| - 2\log T$.*

Proof. Since $T > 1$, $L(s) \cap R(s) = \emptyset$ by definition. Let $t' \in \mathcal{T}$ be the smallest element which is larger than $s/\sqrt{T}$ and $t'' \in \mathcal{T}$ be the largest element smaller than $s\sqrt{T}$. By definition, we have that $\mathcal{T} \setminus (L(s) \cup R(s)) = [t', t''] \cap \mathcal{T}$. Note also that $t''/t' \leq T$, and thus $|[t', t''] \cap \mathcal{T}| \leq \log T + 1 \leq 2\log T$ since $T \geq 10$, which completes the proof. ◀

3.1 Hardness Reduction

The hardness reduction is from a Label-Cover instance $\mathcal{L}(U, V, E \subseteq V \times U, M, N, \{\pi_{vu} : [M] \to [N]\}_{e=(v,u)\in E})$. We shall use $\mathcal{T}$ as defined above for some large enough choice of T depending on δ and ℓ. Note that T is a constant compared to $|V|$ which is an increasing value. The underlying space of the vectors is $\mathcal{X} = \{0,1\}^{V\times[M]}$ i.e., a vector $\mathbf{x} \in \mathcal{X}$ is given by $\mathbf{x} = (x_{v,i})_{v\in V, i\in[M]}$. The reduction yields a distribution $\mathcal{D}_{\mathcal{B}}$ over 2-sized bags and all bags B in its support have the label proportion $\sigma = 1/2$. A random bag from $\mathcal{D}_{\mathcal{B}}$ is given by the following steps:

1. Sample t uniformly at random from $\mathcal{T}$.
2. U.a.r. sample a vertex $u \in U$.
3. Independently and u.a.r. sample vertices $V_u^x = \{\hat{v}_1, \ldots, \hat{v}_t\}$ and $V_u^z = \{\tilde{v}_1, \ldots, \tilde{v}_t\}$ from the neighborhood $N(u)$ of u in V.
4. Randomly sample $J \subseteq [N]$, and let $\overline{J} = [N] \setminus J$.
5. Define a point $\mathbf{x} \in \mathcal{X}$ as follows. For each $i \in [M]$ and $v \in V$ set:

$$x_{v,i} = \begin{cases} 1 & \text{if } v \in V_u^x \text{ and } \pi_{vu}(i) \in J \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

6. Define another point $\mathbf{z} \in \mathcal{X}$ as follows. For each $i \in [M]$ and $v \in V$ set:

$$z_{v,i} = \begin{cases} 1 & \text{if } v \in V_u^z \text{ and } \pi_{vu}(i) \in \overline{J} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

7. Output $(B = \{\mathbf{x}, \mathbf{z}\}, \sigma_B = 1/2)$.

In particular, observe that the points $\mathbf{x}$ and $\mathbf{z}$ are zero outside of the coordinates corresponding to the vertices in $V_u^x \cup V_u^z$.

YES Case. Consider a labeling ρ to the vertices of $\mathcal{L}$ that satisfy all the edges. Define the OR, $h^*(\mathbf{x}) = \bigvee_{v\in V} x_{v,\rho(v)}$. Let u be the choice in Step 2 above, and assume that $\rho(u) \in J$ as chosen in Step 4. Then, we know that for all $v \in V_u^x \cup V_u^z$ (as chosen in Step 3), $\pi_{vu}(\rho(v)) = \rho(u) \in J$. By construction of $\mathbf{x}$ and $\mathbf{z}$ therefore, $h^*(\mathbf{x}) = 1$ and $h^*(\mathbf{z}) = 0$, and thus B is satisfied by h^*. Similarly, when $\rho(u) \in \overline{J}$, we obtain that $h^*(\mathbf{x}) = 0$ and $h^*(\mathbf{z}) = 1$.

3.2 NO Case

Assume for a contradiction an ℓ-clause CNF formula h' s.t. $\Pr_{B \leftarrow \mathcal{D}_{\mathcal{B}}}[h' \text{ satisfies } B] \geq \delta$. From the bi-regularity of $\mathcal{L}$, if any clause of h' contains a negated literal then with probability at least $1 - 2t/|V|$ the literal's coordinate is not from those of the vertices in $V_u^x \cup V_u^z$ and the clause evaluates to 1 on both points of a random bag $B \leftarrow \mathcal{D}_{\mathcal{B}}$. Removing all such clauses we obtain r-clause CNF $h = C_1 \wedge \cdots \wedge C_r$ ($r \leq \ell$) that satisfies at least $\delta - 2t\ell/|V| \geq \delta/2$ fraction of the bags (since we can take $|V| \gg 2^{T+2}\ell/\delta$ as $|V|$ is super-constant). We have the following lemma.

▶ **Lemma 10.** *For any constant $\zeta > 0$, there is a choice of $T = T(\zeta)$ s.t. for any C_i ($i \in [r]$), $\Pr_{B=(\mathbf{x},\mathbf{z}) \leftarrow \mathcal{D}_{\mathcal{B}}}[C_i(\mathbf{x}) \neq C_i(\mathbf{z})] \leq \zeta$.*

Using the above lemma, the NO case proof can be completed by taking $\zeta = \delta/(6\ell)$. By a union bound, the probability that any one of $C_1, \ldots, C_r$ evaluates differently on $\mathbf{x}$ and $\mathbf{y}$ is at most $\delta/6$. This also upper bounds the probability of satisfying the bags of $\mathcal{D}_{\mathcal{B}}$, contradicting our assumption.

Proof of Lemma 10. Fix any clause $C \in \{C_1, \ldots, C_r\}$, and from our construction above C has no negated literals. Call a vertex which has at least one variable from C as *non-empty*, otherwise call it *empty*. For each non-empty vertex v arbitrarily choose i_v such that the (v, i_v)-th variable is in C. For each $u \in U$ let $\mu(u)$ denote the fraction of its neighbors which are non-empty. In our analysis below we shall be collecting the *error* probabilities using

$$\Pr[A] \leq \Pr[A \cap B] + \Pr\left[\bar{B}\right], \text{ and } \Pr[A \cap B] \leq \min\{\Pr[A], \Pr[A \mid B]\} \tag{4}$$

for any two events A and B, where $\bar{B}$ denotes the complement of B.

We will first bound γ which we define to be the probability over the choice of t, u, V_u^x and V_u^z that there is a pair of non-empty vertices $v, v' \in V_u^x \cup V_u^z$ s.t. $\pi_{vu}(i_v) = \pi_{v'u}(i_{v'})$. We call this event Ψ. In this case we can construct a randomized partial labeling ρ for the vertices of $\mathcal{L}$ as follows: for each $v \in V$ which is non-empty, assign it the label i_v defined above. For each u, select a random neighbor v_u and assign u the label $\pi_{v_u u}(i_{v_u})$ if v_u is non-empty. Since $t \leq 2^T$ we obtain that this randomized labeling satisfies in expectation at least $\max_t \gamma/(2t)^2 = \gamma/\left(2^{2(T+1)}\right)$ fraction of the edges. Choosing the soundness of $\mathcal{L}$ to be small enough one can ensure that $\gamma \leq \zeta/100$.

We consider two the cases for t and u below, and in each of them (setting $B = \Psi$ in (4)) we can assume that Ψ does not occur, while incurring at most $\zeta/100$ probability error.

Case I: $t \in L(1/\mu(u))$. In this case the probability that there is non-empty vertex in $V_u^x \cup V_u^z$ is at most $2t\mu(u) \leq 2\left(1/\sqrt{T}\right)(1/\mu(u))\,\mu(u) \leq 2/\sqrt{T}$. Therefore, except with probability at most $2/\sqrt{T}$, C evaluates to zero on $\mathbf{x}$ and $\mathbf{z}$.

Case II: $t \in R(1/\mu(u))$. We shall first show that w.h.p. V_u^x contains a significant number of non-empty vertices, and given that happens, w.h.p. C evaluates to 1 on $\mathbf{x}$. The expected number of non-empty vertices in V_u^x is $t\mu(u) \geq \left(\sqrt{T}/\mu(u)\right)\mu(u) \geq \sqrt{T}$. Therefore, by the Chernoff bound (see Appendix A), except with probability $\exp\left(-\sqrt{T}/8\right)$, the number of non-empty vertices is at least $\sqrt{T}/2$. From the assumption that Ψ does not occur, for all pairs of non-empty vertices $v, v' \in V_u^x$, $\pi_{vu}(i_v) \neq \pi_{v'u}(i_{v'})$. Thus, each $\{x_{v,i_v} \mid v \in V_u^x, v \text{ is non-empty}\}$ is independently set to 1 with probability 1/2. In particular, except with probability $(1/2)^{\sqrt{T}/2}$, at least one of $\{x_{v,i_v} \mid v \in V_u^x, v \text{ is non-empty}\}$ is set to 1 and thus C evaluates to 1 on $\mathbf{x}$. The same argument as above also works for V_u^z and $\mathbf{z}$.

From the analysis of the above two cases, using Lemma 9 and repeated applications of (4) to add up the error probabilities above (for V_u^z and $\mathbf{z}$ as well in Case II) we obtain that:

$$\Pr\left[C(\mathbf{x}) \neq C(\mathbf{z})\right] \leq 2/\sqrt{T} + \zeta/100 + 2\left(\frac{\log T}{T} + \exp\left(-\sqrt{T}/8\right) + 2^{-\sqrt{T}/2}\right) \tag{5}$$

Choosing T to be $100/\zeta^2$ we can ensure that the above probability is at most ζ. ◀

4 Hardness of LLP Learning OR using ℓ-DNF

This section proves the following hardness reduction which implies Theorem 2.

▶ **Theorem 11.** *For any constants $\delta > 0$ and $\ell \in \mathbb{Z}^+$, there is a polynomial time reduction from an instance $\mathcal{L}$ of* Label-Cover *to an* LLP-OR[2] *instance $\mathcal{B}$ s.t.*
YES Case: *If $\mathcal{L}$ is YES instance then there is an* OR *consistent with all the bags of $\mathcal{B}$.*
NO Case: *If $\mathcal{L}$ is a NO instance then there is no ℓ-*DNF *formula satisfying at least $(1/2+\delta)$-fraction of the bags.*

4.1 Hardness Reduction

The setup is the same as in the previous section. The reduction outputs a distribution $\mathcal{D}_{\mathcal{B}}$ over two types of bags with equal probability: (i) the first type has size 1 with label proportion 1, (ii) the second type has size 2 and label proportion 1/2. With T being a large enough constant to be chosen later, the following steps define a random bag of $\mathcal{D}_{\mathcal{B}}$.

1. U.a.r. sample a vertex $u \in U$.
2. With probability 1/2 do the following:
   a. U.a.r. sample vertex $v \in N(u)$ and create the point $\overline{\mathbf{x}}^{(v)}$ as follows: set all coordinates $\{\overline{x}_{v,i}^{(v)}\}_{i=1}^{M}$ to 1, and set all the other coordinates to 0.
   b. Output $\left(B = \{\overline{\mathbf{x}}^{(v)}\}, \sigma_B = 1\right)$.
3. With the remaining probability 1/2 do the following:
   a. Independently and u.a.r. sample vertices $V_u^x = \{\hat{v}_1, \ldots, \hat{v}_T\}$ and $V_u^z = \{\tilde{v}_1, \ldots, \tilde{v}_T\}$ from the neighborhood $N(u)$ of u in V.
   b. Randomly sample $J \subseteq [N]$, and let $\overline{J} = [N] \setminus J$.
   c. Define a point $\mathbf{x} \in \mathcal{X}$ as follows. For each $i \in [M]$ and $v \in V$ set:

$$x_{v,i} = \begin{cases} 1 & \text{if } v \in V_u^x \text{ and } \pi_{vu}(i) \in J \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

   d. Define another point $\mathbf{z} \in \mathcal{X}$ as follows. For each $i \in [M]$ and $v \in V$ set:

$$z_{v,i} = \begin{cases} 1 & \text{if } v \in V_u^z \text{ and } \pi_{vu}(i) \in \overline{J} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

   e. Output $(B = \{\mathbf{x}, \mathbf{z}\}, \sigma_B = 1/2)$.

YES Case. This is easy to see using the same OR formula h^* defined in the previous section. Using the same arguments h^* satisfies all the bags of size 2 from $\mathcal{D}_{\mathcal{B}}$. Further, h^* has exactly one (positive) literal from the coordinates corresponding to each $v \in V$ so that $h^*(\overline{\mathbf{x}}^{(v)}) = 1$ where $\overline{\mathbf{x}}^{(v)}$ is as defined in Step 2a. of $\mathcal{D}_{\mathcal{B}}$. Thus, h^* satisfies all the bags of size 1 as well.

4.2 NO Case

Let us assume that there is an ℓ-DNF h that satisfies $1/2+\delta$ fraction of the bags of $\mathcal{D}_{\mathcal{B}}$. First, observe that if h has a term consisting only of negated literals, then from the bi-regularity of $\mathcal{L}$ and a union bound that term will not have any coordinate from among the vertices chosen in Step 3a with probability at least $1-2T\ell/|V|$. Thus, h will have label proportion 1 on at least $1-2T\ell/|V|$ fraction of the 2-sized bags i.e., h will not satisfy them, implying that the maximum fraction of bags satisfied by h is $1/2+T\ell/|V|$. This is a contradiction since $|V|=\omega(T\ell)$ and can be taken to be large enough. Thus, we may assume that h does not have a term of only negated literals.

Before proceeding, let us call a $v \in V$ as *non-empty* if h has a term in which all the positive literals correspond to v and none of the negated literals correspond to v. Let $\Gamma(v)$ be an arbitrary map from each non-empty v to one such term corresponding to it. Note that $\Gamma(v)$ is injective. Further, define $\Delta(v)$ to be the set of all indices $i \in [M]$ such that the positive literal corresponding to (v,i) occurs in $\Gamma(v)$.

Define for each $u \in U$, κ_u to be the probability that the choice of V_u^x and V_u^z in Step 3a satisfies that there exists a pair $v, v' \in V_u^x \cup V_u^z$ of non-empty vertices such that $\pi_{vu}(\Delta(v)) \cap \pi_{v'u}(\Delta(v')) \neq \emptyset$ (call these *intersecting non-empty* pair of vertices). Using this, let us define a randomized labeling ρ for the vertices in $\mathcal{L}$ as follows: for each non-empty $v \in V$, $\rho(v)$ is chosen u.a.r from $\Delta(v)$, and for each $u \in U$, v_u is chosen u.a.r. from $N(u)$ and then if v_u is non-empty $\rho(u)$ set to i which is chosen u.a.r. from $\Delta(v_u)$. By standard arguments, this labeling satisfies in expectation at least $\kappa_u/(4\ell^2T^2)$ fraction of the edges incident on u, and thus overall in expectation $\mathbb{E}_u[\kappa_u]/(4\ell^2T^2)$ fraction of edges of $\mathcal{L}$. Choosing the soundness of $\mathcal{L}$ to be small enough (and since T is a constant), we can assume that $\mathbb{E}_u[\kappa_u] \leq \delta^2/200$.

By averaging, for at least $\delta/2$ fraction of $u \in U$ (call them *good*) h satisfies at least $1/2+\delta/2$ fraction of the bags from $\mathcal{D}_{\mathcal{B}}\mid_u$ i.e., given the choice of u in Step 1. In particular from the above,

$$\mathbb{E}_u[\kappa_u | u \text{ good }] \leq \delta/100. \tag{8}$$

For any such good u, h must satisfy at least δ fraction of the bags of size 1 and label proportion 1 (as they constitute exactly half of the bags), implying that for at least δ-fraction of $v \in N(v)$, $h\left(\overline{\mathbf{x}}^{(v)}\right)=1$. Clearly, these vertices are non-empty for h to evaluate to 1 on them, and thus any good u has at least δ-fraction non-empty v in $N(u)$.

Let us fix on one such good u. First, from above we can assume by adding an error probability of κ_u that in Step 3a, $V_u^x \cup V_u^z$ does not contain an intersecting non-empty pair. Further, the probability that any term in $\{\Gamma(v) \mid v \in V_u^x \cup V_u^z \text{ and non-empty}\}$ contains a negated literal corresponding to vertices in $V_u^x \cup V_u^z$ is at most $2T\ell/d_U$ where d_U is the uniform degree on U. Since d_U can be taken to be an arbitrarily large constant, possibly by replicating V, we can assume by adding an error probability of $\delta/100$ that this event does not occur.

Given the above, we will show that w.h.p over the choice of a 2-sized bag, h evaluates to 1 on both $\mathbf{x}$ and $\mathbf{z}$. We shall prove this for $\mathbf{x}$, the argument for $\mathbf{z}$ is analogous and the conjunction is obtained by a union bound. Since u is good, the expected number of non-empty vertices in V_u^x is at least δT. By Chernoff bound, by adding an error probability of $\exp(-\delta T/8)$ we can assume there are at least $\delta T/2$ non-empty vertices in V_u^x. By our assumptions above, each term in $\{\Gamma(v) \mid v \in V_u^x \cup V_u^z \text{ and non-empty}\}$ independently evaluates to 1 w.p. at least $(1/2)^\ell$ over the choice of $\mathbf{x}$. Thus, the probability that none of them evaluate to 1 is at most $(1-(1/2)^\ell)^{\delta T/2}$. This analysis can be repeated for V_u^z and $\mathbf{z}$.

Summing up the error probabilities (using repeated applications of (4)), we obtain that:

$$\Pr\left[h(\mathbf{x}) \neq h(\mathbf{z})\right] \leq \kappa_u + \delta/100 + 2\left(\exp(-\delta T/8) + (1-(1/2)^\ell)^{\delta T/2}\right) \leq \kappa_u + \delta/50 \qquad (9)$$

for a good u, using an appropriate choice of $T = O\left(2^\ell \log(1/\delta)/\delta\right)$. By (8), the average of the LHS of (9) over all good u is at most $3\delta/100$, which means that h satisfies on an average at most $1/2 + 3\delta/100$ bags corresponding to a random choice of good u. This is a contradiction to the definition of good u thus completes the NO case analysis.

5 Hardness of LLP Learning Parities

This section is devoted to proving the following hardness reduction which, along with the inapproximability of Smooth-Label-Cover (Th. 7) proves Theorem 3.

▶ **Theorem 12.** *For any constants $\delta > 0$ and $q \in \mathbb{Z}^+$, $q \geq 2$, there is a polynomial time reduction from an instance $\mathcal{L}$ of* Smooth-Label-Cover *to an* LLP-PARITY$[q]$ *instance $\mathcal{B}$ s.t.*
YES Case: *If $\mathcal{L}$ is YES instance then there is a parity consistent with all the bags of $\mathcal{B}$.*
NO Case: *If $\mathcal{L}$ is a NO instance then there is no parity satisfying at least $(q/2^{q-1}+\delta)$-fraction of the bags.*

We begin with the dictatorship test below using which the hardness reduction is described and analyzed in Sec. 5.2.

5.1 Dictatorship Test

Consider a large $M \in \mathbb{Z}^+$, and the space of vectors $\mathbb{F}_2^M$. For some integer $q > 1$, let he dictatorship test distribution $\mathcal{D}^{\mathsf{dict}}_{M,q}$ on (B, σ) be as follows:

1. Choose $B = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}\right\}$ where for each $i \in [M]$, $\left(x_i^{(1)}, \ldots, x_i^{(q)}\right)$ is sampled u.a.r. from $\left\{\mathbf{e}^{(j)}\right\}_{j=1}^q$ where $\mathbf{e}^{(j)} \in \mathbb{F}_2^q$ is the jth coordinate vector.
2. Output $(B, \sigma = 1/q)$.

We prove the following lemma summarizing the completeness and soundness of t $\mathcal{D}^{\mathsf{dict}}_{M,q}$.

▶ **Lemma 13.** *The distribution $\mathcal{D}^{\mathsf{dict}}_{M,q}$ satisfies the following properties:*
Completeness: *For any $i \in [M]$ the parity function $h_i(\mathbf{x}) := x_i$ has the property that $\sigma(B, h_i) = 1/q$ for any B in the support of $\mathcal{D}^{\mathsf{dict}}_{M,q}$ i.e., h_i satisfies all bags of $\mathcal{D}^{\mathsf{dict}}_{M,q}$.*
Soundness: *Let $h(\mathbf{x}) := c_0 \oplus \bigoplus_{i=1}^M c_i x_i$ be such that $|\{i \in [M] \mid c_i = 1\}| = K \leq M$. Then, h satisfies a random $(B, \sigma) \sim \mathcal{D}^{\mathsf{dict}}_{M,q}$ with probability at most $q/2^{q-1} + \exp\left(-2K/q + q/2\right)$.*

Proof. The completeness follows from construction, since for any $(B, \sigma) \sim \mathcal{D}^{\mathsf{dict}}_{M,q}$, where $B = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}\right\}$, $\left(h_i\left(x_i^{(j)}\right)\right)_{j=1}^q = \left(x_i^{(j)}\right)_{j=1}^q \in \left\{\mathbf{e}^{(j)}\right\}_{j=1}^q$ for all $i \in [M]$. Therefore, $\sigma(B, h_i) = 1/q = \sigma$, and therefore h_i satisfies all the bags, for all $i \in [M]$.

The proof of the soundness is given next in Sec. 5.1.1. ◀

5.1.1 Soundness of $\mathcal{D}^{\mathsf{dict}}_{M,q}$

Let $I_h = \{i \in [M] \mid c_i = 1\}$ so that $|I_h| = K$. Letting $B = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}\right\}$ be a random bag from $\mathcal{D}^{\mathsf{dict}}_{M,q}$, for convenience define the random variable $Z_j := h\left(\mathbf{x}^{(j)}\right)$ for $j \in [q]$ and let $\mathcal{D}_Z$ be the distribution on $(Z_1, \ldots, Z_q)$. Since any (B, σ) in the support of $\mathcal{D}^{\mathsf{dict}}_{M,q}$ has $\sigma = 1/q$, using the construction of $B = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}\right\}$ we obtain

$$\begin{aligned}\bigoplus_{j=1}^{q} Z_j &= \bigoplus_{j=1}^{q} h\left(\mathbf{x}^{(j)}\right) = \bigoplus_{j=1}^{q}\left(c_0 \oplus \bigoplus_{i=1}^{M} c_i x_i^{(j)}\right) = \bigoplus_{j=1}^{q} c_0 \oplus \bigoplus_{i\in I_h}\bigoplus_{j=1}^{q} x_i^{(j)} \\ &= \bigoplus_{j=1}^{q} c_0 \oplus \bigoplus_{i\in I_h} 1 = \bigoplus_{j=1}^{q} c_0 \oplus \bigoplus_{i=1}^{K} 1 = \psi^* \in \mathbb{F}_2 \end{aligned} \tag{10}$$

Our goal is to show that $\mathcal{D}_Z$ is close to $\overline{\mathcal{D}}_q$ which we define to be the uniform distribution over the elements of $\mathbb{F}_2^q$ with parity ψ^*. Towards this we prove the following lemma, which shows that the distribution $\mathcal{D}_Z$ has low-bias.

▶ **Lemma 14.** *Consider any strict non-empty subset $S \subsetneq [q]$, s.t. $1 \le |S| = s \le q$. Then,*

$$\left|\Pr\left[\bigoplus_{j\in S} Z_j = 0\right] - \frac{1}{2}\right| \le \frac{1}{2}\cdot \exp\left(-2K/q\right).$$

Proof. First, we may assume that $s \le q/2$, otherwise we can use $[q] \setminus S$ along with (10) to complete the argument. Analogous to (10) we have that

$$\bigoplus_{j\in S} Z_j = \bigoplus_{j\in S} h\left(\mathbf{x}^{(j)}\right) = \bigoplus_{j=1}^{|S|} c_0 \oplus \bigoplus_{i\in I_h}\bigoplus_{j\in S} x_i^{(j)} = \psi_{S,c_0} \oplus \bigoplus_{i\in I_h} r_i \tag{11}$$

where $\psi_{S,c_0} = \bigoplus_{j=1}^{|S|} c_0$ is a constant and $r_i := \bigoplus_{j\in S} x_i^{(j)}$. From the construction of the random bag B, we have that $\{r_i\}_{i\in I_h}$ are iid $\mathbb{F}_2$-valued random variables such that $\Pr[r_i = 1] = s/q, \forall i \in I_h$. In other words, the RHS of (11) denotes the parity of K such iid random variables. To analyze this, let us consider an alternate way of sampling $\{r_i\}_{i=1}^{M}$:

1. Sample $T \subseteq I_h$ by including each $i \in I_h$ into T independently with probability $2s/q \le 1$.
2. For each $i \in I_h \setminus T$, set $r_i = 0$. Independently for each $i \in T$, set $r_i = 1$ w.p. $1/2$ and to 0 otherwise.

It is easy to see that conditioned on $T \ne \emptyset$, $\oplus_{i\in I_h} r_i$ is unbiased. This, along with (11) leads us to,

$$\Pr\left[\bigoplus_{j\in S} Z_j = \psi_{S,c_0}\right] = \frac{1}{2}\cdot\left(1 - \Pr[T=\emptyset]\right) + p\cdot\Pr[T=\emptyset] = \frac{1}{2} + \Pr[T=\emptyset]\left(p - \frac{1}{2}\right) \tag{12}$$

where $p \in [0,1]$ is some probability. Further,

$$\Pr[T=\emptyset] = \left(1 - \frac{2s}{q}\right)^K \le \exp(-2Ks/q)$$

Since $s \ge 1$ and $|p - 1/2| \le 1/2$, the above along with (12) completes the proof. ◀

The rest of the argument is similar to the Vazirani XOR Lemma except we need to show closeness to $\overline{\mathcal{D}}_q$ rather than the uniform distribution. We now transition to Fourier analysis of $\{0,1\}$-valued functions over $\{-1,1\}^q$. For this purpose, we shall map $\mathbb{F}_2$ to $\{-1,1\}$ via $b \mapsto (-1)^b$ and think of $\mathcal{D}_Z$ and $\overline{\mathcal{D}}_q$ as distributions over $\{-1,1\}^q$. First, from the definitions of $\mathcal{D}_Z$ and $\overline{\mathcal{D}}_q$, $\chi_{[q]}(\mathbf{z}) = (-1)^{\psi^*}$ for any $\mathbf{z}$ in the support of $\mathcal{D}_Z$ or $\overline{\mathcal{D}}_q$. Thus,

$$\mathbb{E}_{\mathbf{z}\leftarrow\overline{\mathcal{D}}_q}\left[\chi_{[q]}(\mathbf{z})\right] = \mathbb{E}_{\mathbf{z}\leftarrow\mathcal{D}_Z}\left[\chi_{[q]}(\mathbf{z})\right] = (-1)^{\psi^*} \tag{13}$$

and it is also easy to observe that for any $S \subsetneq [q]$,

$$\mathbb{E}_{\mathbf{z} \leftarrow \overline{\mathcal{D}}_q}[\chi_S(\mathbf{z})] = 0 \quad \text{and} \quad |\mathbb{E}_{\mathbf{z} \leftarrow \mathcal{D}_Z}[\chi_S(\mathbf{z})]| \leq \exp(-2K/q), \tag{14}$$

where the upper bound follows from Lemma 14. Consider any function $f : \{-1,1\}^q \to [0,1]$ having Fourier expansion $\sum_{S \subseteq [q]} \hat{f}_S \chi_S$. Using (13), and (14) we obtain

$$\begin{aligned} \left| \mathbb{E}_{\overline{\mathcal{D}}_q}[f] - \mathbb{E}_{\mathcal{D}_Z}[f] \right| &\leq \exp(-2K/q) \sum_{S \subsetneq [q]} \left| \hat{f}_S \right| \\ &\leq 2^{q/2} \cdot \exp(-2K/q) \sqrt{\sum_{S \subsetneq [q]} \hat{f}_S^2} \leq 2^{q/2} \cdot \exp(-2K/q), \end{aligned}$$

where we use Cauchy-Schwarz and Parseval's bound. We can take f to be indicator function of the event that exactly one of the coordinates is -1. This function evaluates to 1 on $\overline{\mathcal{D}}_q$ with probability exactly $q/2^{q-1}$. Using this along with the above bound completes the proof.

5.2 Hardness Reduction

Our hardness reduction is from an instance $\mathcal{L}$ of Smooth-Label-Cover given in Theorem 7.

Points, bags and label proportions. The initial set of points is defined in the space $\mathbb{F}_2^{V \times [M]}$. For a point $\hat{\mathbf{x}} \in \mathbb{F}_2^{V \times [M]}$ let $\hat{\mathbf{x}}[v] = (\hat{x}_{v,1}, \ldots, \hat{x}_{v,M})$ be the vector of M coordinates corresponding to $v \in V$. Let $\mathcal{D}_{\mathcal{B}}$ be the distribution on bags and label proportions given by the following process.

1. Sample $v \in V$ u.a.r.
2. Sample $(B = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}\}, 1/q) \leftarrow \mathcal{D}^{\mathsf{dict}}_{M,q}$.
3. For $j \in [q]$: define $\hat{\mathbf{x}}^{(j)} \in \mathbb{F}_2^{V \times [M]}$ by letting $\hat{\mathbf{x}}^{(j)}[v] = \mathbf{x}^{(j)}[v]$ and for all $v' \neq v$, $\hat{\mathbf{x}}^{(j)}[v'] = \mathbf{0}$.
4. Output $(\hat{B} = \{\hat{\mathbf{x}}^{(1)}, \ldots, \hat{\mathbf{x}}^{(q)}\}, 1/q)$

Folding and projected point-set. For each $e = (v_1, v_2) \in E$ and $j \in [N]$ define the linear constraint $C[e,j]$ over point $\hat{\mathbf{x}} \in \mathbb{F}_2^{V \times [M]}$ as

$$C[e,j] \Leftrightarrow \bigoplus_{i \in \pi^{-1}_{ev_1}(j)} \hat{x}_{v_1 i} = \bigoplus_{i \in \pi^{-1}_{ev_2}(j)} \hat{x}_{v_2 i}. \tag{15}$$

Let $H \subset \mathbb{F}_2^{V \times [M]}$ be the subspace of all the points which satisfy the set of homogeneous linear constraints $\mathcal{C} := \{C[e,j] \mid e \in E, j \in [N]\}$. We let H be the space in which our final instance resides by linearly projecting all points $\hat{\mathbf{x}}$ created in the support of $\mathcal{D}_{\mathcal{B}}$ into points $\overline{\mathbf{x}} \in H$. Since our final instance is represented in a coordinate system corresponding to a linear basis for H, this also forces any solution h to be represented in a basis for H. In particular, h represented in the original space by $h(\hat{\mathbf{x}}) := c_0 \oplus \langle \mathbf{c}, \hat{\mathbf{x}} \rangle$ (where the inner product is over $\mathbb{F}_2$) must obey $\mathbf{c} \in H$. Let $\overline{\mathcal{D}}_{\mathcal{B}}$ be the new distribution on the bags $(\overline{B}, 1/q)$ given by the linear projection of all the points in the bags of $\mathcal{D}_{\mathcal{B}}$ on to H.

5.2.1 YES Case

In this case, there is a labeling $\rho : V \to [M]$ which satisfies all the edges of $\mathcal{L}$. Consider over $\mathbb{F}_2^{V \times [M]}$ the parity $h^*(\hat{\mathbf{x}}) = \bigoplus_{v \in V} \hat{x}_{v,\rho(v)} =: \langle \mathbf{c}^*, \hat{\mathbf{x}} \rangle$. Since, for any edge $e = (v_1, v_2)$, $\pi_{ev_1}(\rho(v_1)) = \pi_{ev_2}(\rho(v_2))$, $\mathbf{c}^* \in H$. Now fix a choice of v in Step 1 of the distribution $\mathcal{D}_{\mathcal{B}}$. Restricted to the coordinates corresponding to v (since the others are set to 0), h^* is simply

$x_{v,\rho(v)}$. We can now directly apply the completeness property of $\mathcal{D}^{\mathsf{dict}}_{M,q}$ in Lemma 13 to obtain that h^* satisfies all the bags given the choice v. Since this holds for all choices of v, h^* satisfies all the bags of $\mathcal{D}_{\mathcal{B}}$.

5.2.2 NO Case

Assume for a contradiction that there is a parity in the space H that satisfies $(q/2^{q-1}+\delta)$-fraction of the bags of $\mathcal{D}_{\mathcal{B}}$. This parity can be written as

$$h(\hat{\mathbf{x}}) = c_0 \oplus \bigoplus_{v\in V}\bigoplus_{i=1}^{M} c_i\hat{x}_i = c_0 \oplus \bigoplus_{v\in V}\langle \mathbf{c}[v], \hat{\mathbf{x}}[v]\rangle, \tag{16}$$

where $\mathbf{c}$ satisfies the constraints $\mathcal{C}$. By averaging there are $\delta/2$ fraction of *good* $v \in V$ such that h satisfies $(q/2^{q-1}+\delta/2)$-fraction of the bags of $\mathcal{D}_{\mathcal{B}}\mid_v$ i.e., $\mathcal{D}_{\mathcal{B}}$ given v is chosen in Step 1. By the weak-expansion property of $\mathcal{L}$ in Theorem 7, the subset of edges E' induced by the good vertices satisfies $|E'| \geq (\delta/2)^2|E|$. Let $S_v := \{i \in [M] \mid c_{v,i} = 1\}$. From the soundness of $\mathcal{D}^{\mathsf{dict}}_{M,q}$ (Lemma 13), we obtain that all good v satisfy

$$|S_v| \leq \Delta := q(\log(2/\delta) + q/2)/2\ . \tag{17}$$

The smoothness of $\mathcal{L}$ implies that for any good $v \in V$, $\Pr_{e\sim v}\left[\pi_{ev}\left(S_v\right) = |S_v|\right] \geq 1 - |S_v|^2/(2Q) \geq 1 - \Delta^2/(2Q)$. Let $E^* = \{e = (v_1, v_2) \in E' \mid \pi_{ev_r}\left(S_{v_r}\right) = |S_{v_r}|, r = 1, 2\}$. Then

$$\frac{|E^*|}{|E|} \geq \zeta := \frac{\delta^2}{4} - \frac{\Delta^2}{Q}. \tag{18}$$

We have the following lemma.

▶ **Lemma 15.** *For any $e = (v_1, v_2) \in E^*$, $\pi_{ev_1}\left(S_{v_1}\right) \cap \pi_{ev_2}\left(S_{v_2}\right) \neq \emptyset$.*

Proof. Since h satisfies at least one bag of $\mathcal{D}\mid_{v_1}$, $\mathbf{c}[v] \neq \mathbf{0}$, and thus $S_{v_1} \neq \emptyset$. Consider any $j \in \pi_{ev_1}\left(S_{v_1}\right)$. From the definition of E^*, $\left|\pi^{-1}_{ev_1}(j) \cap S_{v_1}\right| = 1$. Thus, $\bigoplus_{i\in\pi^{-1}_{ev_1}(j)} \hat{x}_{v_1 i} = 1$ and from (15) and the fact that $\hat{\mathbf{c}}$ satisfies $C[e, j]$ we obtain that $\pi^{-1}_{ev_2}(j) \cap S_{v_2} \neq \emptyset$. ◀

Let ρ be the randomized labeling to the good vertices given by randomly assigning each good $v \in V$ a label chosen u.a.r. from S_v. From Lem. 15, (17) and (18) we obtain that ρ satisfies in expectation at least, $\nu := \zeta/\Delta^2$ fraction of the edges of $\mathcal{L}$. By choosing the parameter Q in Theorem 7 to be large enough we can take $\zeta \geq \delta^2/8$ and then taking taking the parameter R to be be large enough we obtain a contradiction.

6 Approximately LLP Learning Parities

Here we may assume that $q \geq 2$, otherwise all bags are of size 1 and one can use Gaussian elimination to solve for the satisfying parity. Let $\mathcal{B} := \{(B_k, \sigma_k)\}_{k=1}^m$ be an instance of $\mathsf{LLP\text{-}PARITY}[q]$ over $\mathbb{F}_2^d$ such that there is an (unknown) parity that satisfies all the bags of $\mathcal{B}$. We therefore need to solve for the coefficients of the parity.

Let us first define the subsets of bags $\mathcal{B}_a := \{(B, \sigma) \in \mathcal{B} \mid \sigma = a\}$ for $a \in \{0, 1\}$. We call the bags in $\mathcal{B}_0 \cup \mathcal{B}_1$ as *monochromatic* since we know that the vectors in any such bag are either all labeled 0 or all labeled 1. Therefore, one can write a (possibly non-homogeneous) $\mathbb{F}_2$-linear constraint (in the coefficients of the parity) for each vector in any monochromatic

bag. Further, since the label proportion of each bag is given, the parity of labels in each bag is also determined. Thus, we can add these $\mathbb{F}_2$-linear constraints capturing the parity of the labels for each bag.

Since this system of linear equations is feasible (due to the existence of the satisfying parity) one can do Gaussian elimination to obtain a reduced instance $\mathcal{B}'$ of LLP-PARITY$[q]$ which satisfies the following properties:

1. $\mathcal{B}'$ has no monochromatic bags.
2. A subset of the coefficients may be eliminated or assigned a fixed value $\in \mathbb{F}_2$, and the rest are *free*.
3. For any bag $(B, \sigma) \in \mathcal{B}'$, any assignment to the *free* coefficients yields a labeling which satisfies the parity constraint of that bag. In particular, the set of such assignments yields a (possibly affine) subspace of labelings of size at most 2^{t-1} where $t = |B|$. Let us call this subspace of labelings as F_B.

The algorithm outputs a random parity given by a random assignment to the free coefficients. To analyze its performance, let us consider a bag $(B, \sigma) \in \mathcal{B}'$ where $|B| = t \leq q$ and $t\sigma \in \{1, \ldots, t-1\}$ since $\mathcal{B}'$ has no monochromatic bags. By feasibility there exists a vector in F_B which has Hamming weight $t\sigma$. The probability that the bag will be satisfied by a random parity is precisely the probability that a random point in F_B has Hamming weight $t\sigma$. There are two cases:

1. *F_B contains all vectors of Hamming weight $t\sigma$.* Since $t\sigma \in \{1, \ldots, t-1\}$ the number of such vectors is at least t. Since $|F| \leq 2^{t-1}$, the probability that the bag is satisfied is at least $t/2^{t-1} \geq 1/2^{q-2}$ for any positive integers $t \leq q$ and $q \geq 2$.
2. *F_B does not contains all vectors of Hamming weight $t\sigma$.* In this case, F_B is at most $(t-2)$-dimensional and thus $|F_B| \leq 2^{t-2}$. Since F_B does contain one vector of Hamming weight $t\sigma$, the probability that the bag is satisfied is at least $1/2^{t-2} \geq 1/2^{q-2}$.

References

1. S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.
3. S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.
4. D. Barucic and J. Kybic. Fast learning from label proportions with small bags. *CoRR*, abs/2110.03426, 2021. `arXiv:2110.03426`.
5. G. Bortsova, F. Dubost, S. N. Ørting, I. Katramados, L. Hogeweg, L. H. Thomsen, M. M. W. Wille, and M. de Bruijne. Deep learning from label proportions for emphysema quantification. In *MICCAI*, volume 11071 of *Lecture Notes in Computer Science*, pages 768–776. Springer, 2018. `arXiv:1807.08601`.
6. R. I. Busa-Fekete, H. Choi, T. Dick, C. Gentile, and A. M. Medina. Easy learning from label proportions. *arXiv*, 2023. `arXiv:2302.03115`.
7. L. Chen, T. Fu, A. Karbasi, and V. Mirrokni. Learning from aggregated data: Curated bags versus random bags. *arXiv*, 2023. `arXiv:2305.09557`.
8. L. Chen, Z. Huang, and R. Ramakrishnan. Cost-based labeling of groups of mass spectra. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 167–178, 2004.
9. L. M. Dery, B. Nachman, F. Rubbo, and A. Schwartzman. Weakly supervised classification in high energy physics. *Journal of High Energy Physics*, 2017(5):1–11, 2017.
10. V. Feldman, V. Guruswami, P. Raghavendra, and Y. Wu. Agnostic learning of monomials by halfspaces is hard. *SIAM J. Comput.*, 41(6):1558–1590, 2012.

11 S. Ghoshal and R. Saket. Hardness of learning DNFs using halfspaces. In *Proc. STOC*, pages 467–480, 2021.
12 V. Guruswami, P. Raghavendra, R. Saket, and Y. Wu. Bypassing UGC from some optimal geometric inapproximability results. *ACM Trans. Algorithms*, 12(1):6:1–6:25, 2016. URL: `http://eccc.hpi-web.de/report/2010/177`.
13 J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.
14 J. Hernández-González, I. Inza, L. Crisol-Ortíz, M. A. Guembe, M. J. Iñarra, and J. A. Lozano. Fitting the data from embryo implantation prediction: Learning from label proportions. *Statistical methods in medical research*, 27(4):1056–1066, 2018.
15 S. Khot and R. Saket. Hardness of minimizing and learning DNF expressions. In *Proc. FOCS*, pages 231–240, 2008.
16 C. O'Brien, A. Thiagarajan, S. Das, R. Barreto, C. Verma, T. Hsu, J. Neufeld, and J. J. Hunt. Challenges and approaches to privacy preserving post-click conversion prediction. *CoRR*, abs/2201.12666, 2022. `arXiv:2201.12666`.
17 S. N. Ørting, J. Petersen, M. Wille, L. Thomsen, and M. de Bruijne. Quantifying emphysema extent from weakly labeled ct scans of the lungs using label proportions learning. In *The Sixth International Workshop on Pulmonary Image Analysis*, pages 31–42, 2016.
18 R. O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.
19 R. Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.
20 S. Rueping. SVM classifier estimation from group probabilities. In *Proc. ICML*, pages 911–918, 2010.
21 R. Saket. Learnability of linear thresholds from label proportions. In *Proc. NeurIPS*, 2021. URL: `https://openreview.net/forum?id=5BnaKeEwuYk`.
22 R. Saket. Algorithms and hardness for learning linear thresholds from label proportions. In *Proc. NeurIPS*, 2022. URL: `https://openreview.net/forum?id=4LZo68TuF-4`.
23 L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.
24 J. Wojtusiak, K. Irvin, A. Birerdinc, and A. V. Baranova. Using published medical results and non-homogenous data in rule learning. In *Proc. International Conference on Machine Learning and Applications and Workshops*, volume 2, pages 84–89. IEEE, 2011.
25 F. X. Yu, K. Choromanski, S. Kumar, T. Jebara, and S. F. Chang. On learning from label proportions. *CoRR*, abs/1402.5902, 2014. `arXiv:1402.5902`.

A Useful Bounds

We use the Chernoff bound stated as follow.

▶ **Theorem 16** (Chernoff Bound). *Suppose $X_1, \ldots, X_n$ and independent $\{0,1\}$-valued random variables with $S = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[S]$. Then, $\Pr[S \leq (1-\delta)\mu] \leq \exp(-\delta^2\mu/2)$ for any $\delta > 0$.*

Dependency Schemes in CDCL-Based QBF Solving: A Proof-Theoretic Study

Abhimanyu Choudhury
The Institute of Mathematical Sciences, Chennai, India
Homi Bhabha National Institute, Training School Complex, Anushaktinagar, Mumbai, India

Meena Mahajan
The Institute of Mathematical Sciences, Chennai, India
Homi Bhabha National Institute, Training School Complex, Anushaktinagar, Mumbai, India

Abstract

In Quantified Boolean Formulas QBFs, dependency schemes help to detect spurious or superfluous dependencies that are implied by the variable ordering in the quantifier prefix but are not essential for constructing countermodels. This detection can provably shorten refutations in specific proof systems, and is expected to speed up runs of QBF solvers. The proof system QCDCL recently defined by Beyersdorff and Böhm (LMCS 2023) abstracts the reasoning employed by QBF solvers based on conflict-driven clause-learning (CDCL) techniques. We show how to incorporate the use of dependency schemes into this proof system, either in a preprocessing phase, or in the propagations and clause learning, or both. We then show that when the reflexive resolution path dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ is used, a mixed picture emerges: the proof systems that add $\mathtt{D}^{\mathtt{rrs}}$ to QCDCL in these three ways are not only incomparable with each other, but are also incomparable with the basic QCDCL proof system that does not use $\mathtt{D}^{\mathtt{rrs}}$ at all, as well as with several other resolution-based QBF proof systems. A notable fact is that all our separations are achieved through QBFs with bounded quantifier alternation.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases QBF, CDCL, Resolution, Dependency schemes

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.38

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2023/061/

Acknowledgements Part of this work was done when the second author was at the Simons Institute for the Theory of Computing at Berkeley during March–May 2023, participating in the Extended Reunion of the program Satisfiability: Theory, Practice, and Beyond.

1 Introduction

Despite the NP-hardness of propositional satisfiability, SAT solvers today are amazingly efficient in solving real-world instances. The best algorithms solving SAT in practice are based on the paradigm of conflict-driven clause learning CDCL, that revolutionised SAT solving in the nineties. Such algorithms use a generic template as follows: repeatedly decide values of some variables, propagate hard constraints (unit clauses) until a conflict is reached, "learn" a new clause from the conflict, backtrack and continue. For unsatisfiable formulas, the learning process yields a refutation in the proof system Resolution, and it was shown over a decade ago that resolution proofs can themselves be mimicked within this framework, so CDCL equals Resolution, [20, 1]. Hence, a proof-complexity-theoretic analysis of Resolution has revealed deep insights into the strengths and limitations of this CDCL paradigm.

With the success of propositional SAT solvers, there are many ambitious attempts now to tackle more expressive/succinct formalisms. In particular, for the PSPACE-complete problem of deciding the truth of Quantified Boolean Formulas QBF, there are now many

© Abhimanyu Choudhury and Meena Mahajan;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 38; pp. 38:1–38:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

solvers, as well as a rich (and still growing) theory about the underlying formal proof systems. Designing solvers for QBFs is a useful enterprise because many industrial applications seem to lend themselves more naturally to expressions involving both existential and universal quantifiers; see for instance [23, 9].

The proof system Resolution can be lifted to the QBF setting in many ways. The "CDCL way" is to add a universal reduction rule, giving rise to the system Q-Res and the more general QU-Res. Allowing contradictory literals to be merged under certain conditions gives rise to the system long-distance Q-Resolution LDQ-Res.

Another "CDCL" way is to lift the CDCL algorithm itself to a QCDCL algorithm: decide values of variables, usually respecting the order of quantified alternation, propagate unit constraints, interpreting unit modulo universal reductions, repeat until a conflict is reached, learn a new clause, backtrack and continue. For false formulas, the learning process yields a long-distance Q-resolution refutation. However, the QCDCL refutation itself is much more restricted than an LDQ-Res refutation. In [7], a formal proof system QCDCL was abstracted out of the QCDCL algorithm. Noting that potentially the decision policy and the propagation policy could be modified, the authors of [7] actually formalised four different QCDCL-based proof systems. The system underlying most solvers is $\mathtt{QCDCL}_{\mathtt{RED}}^{\mathtt{LEV\text{-}ORD}}$, which we refer to as QCDCL without any sub/super-script; for the other systems we explicitly write the policies.

While the QCDCL proof system explains the correctness of solvers for false QBFs, it ignores cube-learning from satisfying assignments. In practice, cube-learning is essential to the completeness of a QCDCL solver; it is integral to proving a QBF to be true. (A QBF solver algorithm does not know in advance whether the input formula is true or false. It learns clauses and cubes, and concludes false/true when the empty clause/cube has been learnt.) The choice to ignore this was made in [7] because the focus there (as also here) was on refutational proof systems, proving QBFs false. In this setting, cube-learning is not an essential ingredient. However, in [13], the authors defined the system $\mathtt{QCDCL}^{\mathtt{cube}}$ that incorporates cube-learning on top of the original QCDCL, and found that cube-learning was in fact advantageous even in constructing shorter refutations for false QBFs. More recent work (see [12]) in fact redefines QCDCL as the augmented system with cube-learning incorporated. For this paper, we retain the notation from [7]. An interesting outcome from [12], though not directly relevant to our work, is that extracted refutations do not capture the full generality of LDQ-Res.

DepQBF [16] is the leading QCDCL solver and has many versions. Its base version still employs what the authors call "vanilla QCDCL", and its behaviour on false QBFs is explained by the proof system QCDCL which we are interested in exploring. Later versions of DepQBF provide options of turning on "cube learning" (when "turned on", its behaviour is explained by the proof system $\mathtt{QCDCL}^{\mathtt{cube}}$) and also offer heuristics like whether or not to allow "dependency scheme aware propagation" and/or apply "pure literal elimination".

A heuristic that has been found to be useful in many QBF solvers, and has been formalised in proof systems, is to eliminate easily-detectable spurious dependencies. In a prenex QBF, a variable "depends" on the variables preceding it in the quantifier prefix; where "depends" means that a Herbrand/Skolem function for the variable is a function of the preceding variables. However, a Herbrand function or countermodel may not really need to know the values of all preceding variables. A dependency scheme filters out as many of such unnecessary dependencies as it can detect, producing what is in effect a Dependency QBF, DQBF. Although DQBF is a significantly richer formalism that is known to be NEXP-complete (see [2, 22]), these heuristics are not aiming to solve DQBFs in general. Rather, they algorithmically detect spurious dependencies and disregard them as the algorithm

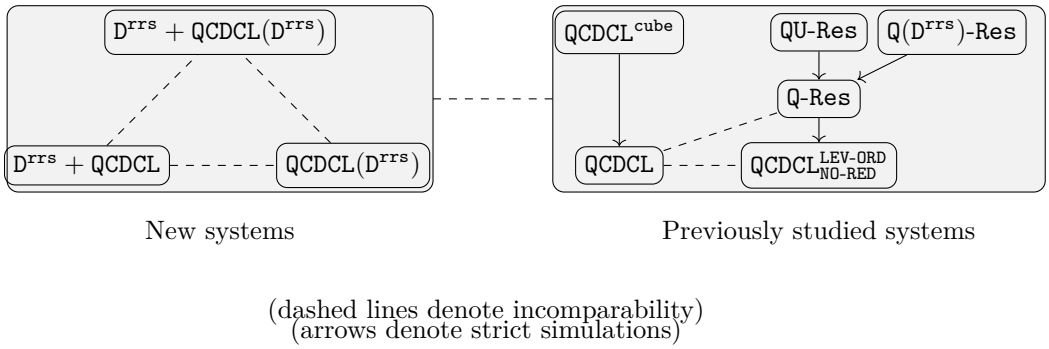


Figure 1 Relations between proof systems.

proceeds. See [15, 21, 16, 17] for early work on this topic. Often the use of a dependency scheme makes the solvers run faster. Now, the universal reduction rule in the proof systems (say in Q-Res, LDQ-Res) can be applied in more settings because there are fewer dependencies, and this can shorten refutations significantly. See for instance [10, 24, 19]. Note that the use of a dependency scheme must be proven to be sound and complete, and this in itself is often quite involved. The notion of a dependency scheme being "normal" was introduced in [19], where it is shown that adding any normal dependency scheme to LDQ-Res preserves soundness and completeness.

In this paper, we examine how the usage of a dependency scheme can affect proof systems underlying the QCDCL algorithm. As far as we are aware, such a theoretical study has not been undertaken before, even though many current QBF solvers are based on the QCDCL paradigm and also do use dependency schemes. Specifically, we focus on the proof system QCDCL (in the notation of [7], the $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$ proof system), underlying most QCDCL-based solvers, and on the dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ which has been studied in the context of Q-Res and LDQ-Res, see [24, 10, 19]. We note that the proof system QCDCL can be made aware of dependency schemes in more than one way. We identify two natural ways: (1) use a dependency scheme D to preprocess the formula, performing reductions in the initial clauses whenever permitted by the scheme, and (2) use a dependency scheme D in the QCDCL algorithm itself, in enabling unit propagations and in learning clauses. Denoting the first way as $\mathtt{D} + \mathtt{QCDCL}$ and the second as $\mathtt{QCDCL}(\mathtt{D})$, and noting that we could even use different dependency schemes in both these ways, we obtain the system $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$. When $\mathtt{D}_1$ and $\mathtt{D}_2$ are both the trivial dependency scheme $\mathtt{D}^{\mathtt{trv}}$ inherited from the linear order of the quantifier prefix, this system is exactly QCDCL.

Our contributions are as follows:

1. We formalise the proof system $\mathtt{D}' + \mathtt{QCDCL}(\mathtt{D})$ for dependency schemes $\mathtt{D}, \mathtt{D}'$, and note that whenever $\mathtt{D}', \mathtt{D}$ are normal schemes, $\mathtt{D}' + \mathtt{QCDCL}(\mathtt{D})$ is sound and complete (Theorem 3.2).
2. For $\mathtt{D}, \mathtt{D}' \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$, we study the four systems $\mathtt{D}' + \mathtt{QCDCL}(\mathtt{D})$. As observed above, one of them is QCDCL itself, while the others are new systems. We compare these systems with each other and show that they are all pairwise incomparable (Theorem 5.1). We also show that each of them is incomparable with each of the systems $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, Q-Res, $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$, and QU-Res(Theorem 5.2), as well as with $\mathtt{QCDCL}^{\mathtt{cube}}$(Theorem 5.3).

   Relations among various proof systems are shown in Figure 1.

In other words, making QCDCL algorithms dependency-aware is a "mixed bag": in some situations this shortens runs while in others it is disadvantageous. Here are our thoughts on what this actually means.

That $\mathtt{QCDCL}(\mathtt{D})$ is stronger than $\mathtt{QCDCL}$ at times is to be expected; after all, that is why the heuristic evolved. That it can be weaker at times appears a bit surprising until one recalls that even when $\mathtt{QCDCL}$ was formalised in [7], it was shown that the no-reduction version $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$ can have an advantage over $\mathtt{QCDCL}$; for some formulas, enabling more reductions and unit propagations can send the trails down into a trap where refuting a hard sub-formula becomes inevitable. Since dependency schemes do exactly this enabling of more reductions and propagations, custom formulas can be designed where the difference is not just between no-reductions and reductions, but also between reductions and dependency-aware reductions. This is a consequence of the level-ordering of decisions and the forcing of all unit propogations with reduction, and may not hold for the other variants of $\mathtt{QCDCL}$.

That $\mathtt{D} + \mathtt{QCDCL}$ can be stronger than $\mathtt{QCDCL}$ is again to be expected. That it can be weaker seems really counter-intuitive, but is again related to the comment above: the preprocessing shortens clauses and thus enables more unit propagations in subsequent trails.

One direction of our separation between $\mathtt{D} + \mathtt{QCDCL}$ and $\mathtt{QCDCL}(\mathtt{D})$ was genuinely surprising to us. We construct formulas where after preprocessing (as in $\mathtt{D} + \mathtt{QCDCL}$) the resulting formula is propositional and easy to refute in Resolution, and hence the original formula is easy to refute in $\mathtt{D} + \mathtt{QCDCL}$. However, the same formula is hard for $\mathtt{QCDCL}(\mathtt{D})$, Section 4.5! In other words, it is not enough for the QCDCL algorithm to be dependency-aware; this awareness must be achieved at the right stage of the algorithm.

The fact that $\mathtt{QCDCL}(\mathtt{D})$, $\mathtt{D} + \mathtt{QCDCL}$, and $\mathtt{D} + \mathtt{QCDCL}(\mathtt{D})$ are all incomparable with $\mathtt{QCDCL}^{\mathtt{cube}}$ is note-worthy and interesting as allowing for cube-learning always adds strength and makes things easier as compared to without cube-learning; $\mathtt{QCDCL}^{\mathtt{cube}}$ as a proof system is known to be strictly more powerful than $\mathtt{QCDCL}$ [13]. Our results show that switching on cube-learning (which most current solvers do by default) and switching on dependency-awareness as proposed here are orthogonal options. Which option is better may depend on the setting from which the instances to be solved arise.

This work is based on formalisms in [24, 10, 19, 7]. See [7] for an extensive bibliography of relevant work.

The rest of this paper is organised as follows. After spelling out notation and required preliminaries in Section 2, including defining dependency schemes and describing the $\mathtt{QCDCL}$ proof system, we show in Section 3 that the addition of normal dependency schemes results in sound and complete proof systems. In Section 4 we present, for some previously studied formulas as well as for some newly designed formulas, lower and/or upper bounds in the $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$ systems when $\mathtt{D}_1, \mathtt{D}_2$ are in $\{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$. Using these bounds, we conclude in Section 5 that these new systems are pairwise incomparable with each other as well as with each of $\mathtt{QCDCL}$, $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, $\mathtt{Q\text{-}Res}$, $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$, $\mathtt{QU\text{-}Res}$, $\mathtt{QCDCL}^{\mathtt{cube}}$. We end with some concluding remarks in Section 6.

Some proofs are briefly sketched due to space constraints; full details can be found in [14].

2 Preliminaries

2.1 Basics

A Quantified Boolean Formula in prenex conjunction normal form (PCNF) consists of a prefix with an ordered list of variables, each quantified either existentitally or universally, and the matrix, which is a set of clauses over these variables. That is, it has the form

$$\Phi = \mathcal{Q}\vec{x} \cdot \varphi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi(x_1, x_2, \ldots, x_n)$$

where φ is a propositional formula in CNF.

The formula is true if there are (Skolem) functions s_i for each existentially quantified variable x_i, where each such s_i depends only on universally quantified variables x_j with $j < i$, such that substituting these s_i in φ yields a tautology. Similarly, the formula is false if there are (Herbrand) functions h_i for each universally quantified variable x_i, where each such h_i depends only on existentially quantified variables x_j with $j < i$, such that substituting h_i in φ yields an unsatisfiable formula.

In this paper, we focus on false formulas; refutations must rule out the existence of Skolem functions. In the proof system `Q-Res`, a refutation of a false QBF is a derivation of the empty clause $\square$ from the clauses in the matrix, using two rules: Resolution (from $A = C \vee x$ and $B = D \vee \neg x$, derive $C \vee D$, provided the pivot x is existential and $C \vee D$ is not tautological. We denote this as $C \vee D = \texttt{res}(A, B, x)$), and Universal Reduction (from $C \vee u$ derive C if u is universal and no existential variable in C appears to the right of u in the prefix). The proof system `QU-Res` generalises `Q-Res` by allowing resolution on universal pivots as well. The proof system `LDQ-Res` generalises `Q-Res` in a different way, allowing the derivation of seemingly-tautological clauses in resolution under certain conditions: a universal variable u appearing in opposite polarities in C and D is represented as the merged literal u^* in the resolvent, provided it is to the right of the pivot x.

A proof system `P` simulates a proof system `Q` if, for every formula, the size of the shortest `P` refutation is polynomial in the size of the shortest `Q` refutation. The systems `QU-Res` and `LDQ-Res` are both strictly more powerful than `Q-Res` and incomparable with each other.

For a set S of clauses and a literal ℓ, we use shorthand $\ell \vee S$ to denote the set of clauses $\{\ell \vee C \mid C \in S\}$.

2.2 Dependency Schemes

In a PCNF formula Φ, for any existential variable x, any witnessing Skolem function $s(x)$ (if such functions exist) can depend only on those universal variables that are quantified to the left of x in the quantifier prefix. The *trivial dependency scheme* $\mathtt{D}^{\mathtt{trv}}$ records this: for a PCNF formula Φ, the relation $\mathtt{D}^{\mathtt{trv}}(\Phi)$ consists of all pairs (u, x) where existential variable x appears to the right of universal variable u in the order of quantifier prefix. If a pair (u, x) appears in $\mathtt{D}^{\mathtt{trv}}(\Phi)$, we say that x depends on u.

A dependency scheme aims to weed out redundant pairs from $\mathtt{D}^{\mathtt{trv}}$ while preserving the (non)-existence of Skolem functions. Given a PCNF formula Φ, it maps each variable x to a set of variables whose values should be sufficient to determine how to set x. The most basic of dependency schemes weeds out nothing and is the trivial dependency scheme $\mathtt{D}^{\mathtt{trv}}$ itself. A non-trivial dependency scheme $\mathtt{D}$ produces, for any formula Φ, a subset $\mathtt{D}(\Phi)$ of the trivial dependencies $\mathtt{D}^{\mathtt{trv}}(\Phi)$; it does not introduce new dependencies. Some non-trivial schemes are the standard scheme $\mathtt{D}^{\mathtt{std}}$ and the reflexive resolution path scheme $\mathtt{D}^{\mathtt{rrs}}$; see [24]. Roughly speaking, in $\mathtt{D}^{\mathtt{rrs}}$, (u, x) is in the dependency relation of a formula if $(u, x) \in \mathtt{D}^{\mathtt{trv}}$ and there is a sequence of clauses with the first containing u, the last containing $\bar{u}$, some intermediate consecutive clauses containing x and $\bar{x}$, and where each pair of consecutive clauses has an existential variable, quantified after u, in opposite polarities. The non-existence of such a sequence implies that if at all there are Skolem functions for x, then there exists a Skolem function for x which does not use information about u; hence x need not depend on u. Formally, the dependence scheme is defined as follows:

▶ **Definition 2.1** (Reflexive Resolution Path Dependency Scheme, [24]). *For a QBF $\Phi = \mathcal{Q}\phi$, the pair (u, x) is in $\mathtt{D}^{\mathtt{rrs}}(\Phi)$ if and only if $(u, x) \in \mathtt{D}^{\mathtt{trv}}(\Phi)$ and there exists a sequence of clauses $C_1, \cdots, C_n \in \phi$ and a sequence of literals $l_1, \cdots, l_{n-1}$ such that:*

- $u \in C_1$ *and* $\bar{u} \in C_n$,
- $x = var(l_i)$ *for some* $i \in [n-1]$,
- $var(l_i) \neq var(l_{i+1})$ *for each* $i \in [n-2]$, *and*
- $(u, var(l_i)) \in \mathtt{D}^{\mathtt{trv}}(\Phi)$, $l_i \in C_i$ *and* $\bar{l}_i \in C_{i+1}$ *for each* $i \in [n-1]$.

For a dependency scheme $\mathtt{D}$, a QBF Φ, a universal literal $\ell_u \in \{u, \bar{u}\}$ and an existential literal $\ell_x \in \{x, \bar{x}\}$, we say that ℓ_x blocks ℓ_u if $(u, x) \in \mathtt{D}(\Phi)$; in particular, this implies that x is quantified after u. For a clause C we denote by $\mathtt{red\text{-}D}(C)$ the subclause obtained by removing all universal literals which are not blocked by any other literal in C. We denote by $\mathtt{red\text{-}D}(\Phi)$ the QBF Ψ obtained by replacing each clause C in the matrix of Φ with the clause $\mathtt{red\text{-}D}(C)$. When $\mathtt{D} = \mathtt{D}^{\mathtt{trv}}$, we use the notation $\mathtt{red}(C)$ and $\mathtt{red}(\Phi)$.

The proof systems Q-Res and LDQ-Res, augmented with a dependency scheme D [19], permit universal reduction of u under the more relaxed requirement that $(u, x) \notin \mathtt{D}$ for any existential variable $x \in C$. That is, they permit the derivation of $\mathtt{red\text{-}D}(C)$ from C.

In a QBF with a leading universal block, universal variables from the first block appear in a LDQ-Res derivation in only one polarity. This feature is useful in proving soundness of LDQ-Res. Soundness of LDQ(D)-Res can be proven in a similar vein, provided the dependency scheme D does not permit derivations violating this property, and provided that applying a partial assignment can possibly erase existing dependencies but cannot create new ones. This leads to the definition of the following important subclass of dependency schemes, the normal dependency schemes.

▶ **Definition 2.2** (Normal Dependency Scheme, [19]). *A dependency scheme* D *is*

- *monotone if for every PCNF formula* ϕ, *and every assignment* τ *to a subset of* $var(\phi)$, $\mathtt{D}(\phi[\tau]) \subseteq \mathtt{D}(\phi)$. *(Here* $\phi[\tau]$ *is the restriction of* ϕ *obtained by applying the partial assignment* τ *to it.)*
- *simple if for every PCNF formula* Φ *of the form* $\Phi = \forall X_1 \exists Y_1 \cdots \forall X_n \exists Y_n \phi$, *every (minimal)* LDQ(D)-Res *derivation* P *of a clause from* Φ, *and every* $u \in X_1$, *either* u *or* $\bar{u}$ *does not appear in* P.
- *normal if it is both monotone and simple.*

Normal dependency schemes are important because if D is normal, then LDQ(D)-Res is a sound proof system [19]. The dependency schemes $\mathtt{D}^{\mathtt{trv}}$, $\mathtt{D}^{\mathtt{std}}$, $\mathtt{D}^{\mathtt{rrs}}$ are all normal.

2.3 The proof system QCDCL

This proof system QCDCL defined in [7] formalises the reasoning in QCDCL algorithms. A refutation of a false QBF is a sequence of triples of the form (T, C, π) where T is a trail (in the QCDCL algorithm) ending in a conflict, C is the clause learnt from this trail, and π is the LDQ-Res derivation of C explaining how C is learnt. (Recall that in (Q)CDCL, a trail is a sequence of literals, some of which are set due to decisions made by the algorithm and the others are propagated literals set due to a unit constraint. Following the standard convention, we denote decision literals in a trail in boldface. A trail thus has the form

$$T := (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

where the literals d_i are decision literals, and the literals $p_{i,j}$ are propagated literals.) From the last triple in the sequence we can learn the empty clause, completing the refutation. Three factors affect the construction of the refutation.

1. The decision policy: how to choose the next variable to branch on. In standard QCDCL (i.e $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$, the focus of this paper), decisions must respect the quantifier prefix level order. (Variables x, y are at the same level if they are quantified the same way, and no variable with a different quantification appears between them in the prefix order.) Other policies such as ASS-ORD, ASS-R-ORD, UNI-ANY, are also possible; see [7, 11].
2. The unit propagation policy. Upon a partial assignment α to some variables, when does a clause C propagate a literal? In the No-Reduction policy, a clause C is unit if exactly one literal ℓ of C is unset, and this literal is propagated. In the Reduction policy, used by most current QCDCL solvers [16, 18], a clause C propagates literal ℓ if after restricting C by α and applying all possible universal reductions, only ℓ remains. In standard QCDCL the Reduction policy is used. Also, propagations are made as soon as possible; see the description of natural trails (Def 3.4 in [7].
   In the notation of [7], for a decision policy P and a propagation policy R, the corresponding QCDCL proof system is denoted $\mathtt{QCDCL}^{P}_{R}$. Thus standard QCDCL is $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$. Other variants are also defined in [7]; in particular $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$.
3. The set of learnable clauses. These explain the conflict at the end of a trail.
   ▶ **Definition 2.3** (learnable clauses). *From a trail*

$$T := (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

ending in a conflict $p_{(r,g_r)} = \square$, the sequence L_T of learnable clauses has a clause associated with each propagation in the trail, and one more clause, described by tracing the conflict backwards through the trail as follows. ($\mathtt{ante}(\ell)$ denotes the clause that causes literal ℓ to be propagated; i.e. the antecedent.)

- $C_{(r,g_r)} = \mathtt{red}(\mathtt{ante}(p_{(r,g_r)}))$.
- *For $i \in \{0, 1, \cdots, r\}$ and $j \in [g_i - 1]$,*

$$C_{(i,j)} = \begin{cases} \mathtt{red}[\mathtt{res}(C_{(i,j+1)}, \mathtt{red}(\mathtt{ante}(p_{(i,j)})), p_{(i,j)})] & \text{if } \bar{p}_{(i,j)} \in C_{(i,j+1)} \\ C_{(i,j+1)} & \text{otherwise} \end{cases}$$

- *For $i \in \{0, 1, \cdots, r-1\}$.*

$$C_{(i,g_i)} = \begin{cases} \mathtt{red}[\mathtt{res}(C_{(i+1,1)}, \mathtt{red}(\mathtt{ante}(p_{(i,g_i)})), p_{(i,g_i)})] & \text{if } \bar{p}_{(i,g_i)} \in C_{(i+1,1)} \\ C_{(i+1,1)} & \text{otherwise} \end{cases}$$

In the above formulation of the QCDCL system, we only consider trails that end in a conflict. Trails ending in a satisfying assignment are ignored. This is enough to ensure refutational completeness – the ability to prove all false QBFs false. From satisfying assignments, solvers can learn cubes (or terms), and this is necessary to prove true QBFs true. In [13] it was shown that allowing cube (or term) learning from satisfying assignments can also be advantageous while refuting false QBFs. This led to the definition of the proof system $\mathtt{QCDCL}^{\mathtt{cube}}$, which was shown to be strictly stronger than the standard QCDCL system i.e. $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$.

Our focus, however, is on adding dependencies to the basic QCDCL system without cube learning, so wherever we talk about QCDCL as a proof system we refer to $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$.

3 Adding Dependency Schemes to the QCDCL proof system

We first describe the generic addition of dependency schemes to QCDCL, and then show soundness and completenes for *normal* schemes. For a decision policy P and a propagation policy R, the corresponding QCDCL proof system is denoted $\mathtt{QCDCL}^{P}_{R}$. Adding a dependency scheme D to this system can affect P, R, as well as the set of learnable clauses.

For the decision policy $P = \texttt{LEV-ORD}$, which is the focus of this work, adding a dependency scheme $\texttt{D}$ does not affect the decision policy.

For the propagation policy, the notion of unit clauses depends on the universal reductions allowed, which in turn is affected by the dependency scheme. When $R = \texttt{NO-RED}$, no universal reductions are allowed anyway, so adding a dependency scheme to the proof system does not affect the policy. When $R = \texttt{RED}$, the definition of a unit propagation changes. A clause C propagates a literal ℓ at a position in the trail if the $\forall(\texttt{D})$-reduction of C restricted to the trail so far is a unit clause. That is, the partial assignment α specified by the trail so far does not satisfy C, and after restricting C by α, applying all universal reductions allowed by $\texttt{D}$ leaves the single literal ℓ; $\texttt{red-D}(C|_\alpha) = \{\ell\}$. We denote this propagation policy as $\texttt{RED} + \texttt{D}$.

The dependency scheme modifies the reduction rule, which modifies the set of learnable clauses. The set of learnable clauses is now defined in a similar way as in Definition 2.3, but replacing $\texttt{red}$ everywhere with $\texttt{red-D}$, the $\forall(\texttt{D})$ rule for universal reduction with respect to the dependency.

A completely different way in which a dependency scheme $\texttt{D}$ can be added to $\texttt{QCDCL}$ proof systems is by adding it as a preprocessing step, by applying the $\texttt{red-D}$ rule on the axioms (clauses of the given formula). That is, produce $\texttt{QCDCL}$ refutations of $\texttt{red-D}(\Phi)$ instead of Φ.

These two ways of adding dependency schemes to $\texttt{QCDCL}$ – (1) in the trail construction, propagation and learning itself, or (2) as pre-processing – can both be combined. For a particular dependency scheme $\texttt{D}$, we can think of three distinct proof systems:

- $\texttt{QCDCL}(\texttt{D})$: use $\texttt{D}$ for unit propagations and learning, but not for preprocessing.
- $\texttt{D} + \texttt{QCDCL}$: use $\texttt{D}$ only to preprocess the formula.
- $\texttt{D} + \texttt{QCDCL}(\texttt{D})$: use $\texttt{D}$ for preprocessing, and also during propagation and learning.

Going a step further, we can even use different dependency schemes in the preprocessing and in the actual trails. Thus, formally, we define the general proof system $\texttt{D}' + \texttt{QCDCL}(\texttt{D})$:

▶ **Definition 3.1** ($\texttt{D}' + \texttt{QCDCL}(\texttt{D})$ proof system).
For a false QBF $\Psi = \mathcal{Q} \cdot \psi$ and a dependency scheme $\texttt{D}$, a $\texttt{QCDCL}(\texttt{D})$ derivation of a clause C from Ψ is a sequence of triples (T_i, C_i, π_i), or equivalently, a triple of sequences

$$\iota := ((T_1, \cdots, T_m), (C_1, \cdots, C_m), (\pi_1, \cdots, \pi_m))$$

where for each $i \in [m]$, the trail T_i follows policies $\texttt{LEV-ORD}$ and $\texttt{RED} + \texttt{D}$, each clause $C_j \in L_{T_j}$ is a clause learnable from T_j using the $\texttt{red-D}$ rule, and $C_m = C$. Each π_i is the derivation of C_i from $\mathcal{Q} \cdot (\psi \cup \{C_1, \cdots, C_{i-1}\})$ in $\texttt{LDQ}(\texttt{D})\texttt{-Res}$.

For a false QBF $\Phi = \mathcal{Q} \cdot \phi$ and dependency schemes $\texttt{D}, \texttt{D}'$, a $\texttt{D}' + \texttt{QCDCL}(\texttt{D})$ deriviation of a clause C from Φ is a $\texttt{QCDCL}(\texttt{D})$ derivation of C from $\Psi = \texttt{red-D}'(\Phi)$.

If $C = (\square)$, then the derivation ι is called a refutation.

Note that $\texttt{QCDCL}$ is exactly the proof system $\texttt{D}^{\texttt{trv}} + \texttt{QCDCL}(\texttt{D}^{\texttt{trv}})$. Using other dependency schemes instead of $\texttt{D}^{\texttt{trv}}$ is a natural generalisation.

We now show that adding normal dependency schemes $\texttt{D}_1, \texttt{D}_2$ preserves soundness and completness. (The main focus of this work is the cases when $\texttt{D}_1, \texttt{D}_2$ are either $\texttt{D}^{\texttt{rrs}}$ or $\texttt{D}^{\texttt{trv}}$. However, the proofs we present below do not become any simpler for these special cases.)

▶ **Theorem 3.2.** *If $\texttt{D}_1$ and $\texttt{D}_2$ are normal dependency schemes, then $\texttt{D}_1 + \texttt{QCDCL}(\texttt{D}_2)$ is a sound and complete proof system.*

Proof. Soundness: Let ι be a $\texttt{D}_1 + \texttt{QCDCL}(\texttt{D}_2)$ refutation of a QBF Φ. By definition, this is a $\texttt{QCDCL}(\texttt{D}_2)$ refutation of the QBF $\Psi = \texttt{red-D}_1(\Phi)$. Now, every $\texttt{QCDCL}(\texttt{D})$ refutation has an underlying $\texttt{LDQ}(\texttt{D})\texttt{-Res}$ refutation. Thus from ι we can extract a $\texttt{LDQ}(\texttt{D}_2)\texttt{-Res}$ refutation Π of

Ψ. Since $\mathtt{D}_2$ is a normal dependency scheme, $\mathtt{LDQ}(\mathtt{D}_2)$-$\mathtt{Res}$ is a sound proof system [19], and therefore Ψ is a false QBF. By completeness of $\mathtt{LDQ}$-$\mathtt{Res}$, there exists a $\mathtt{LDQ}$-$\mathtt{Res}$ refutation Π' of Ψ. The reductions made to obtain Ψ from Φ, followed by the derivation steps in Π', gives a $\mathtt{LDQ}(\mathtt{D}_1)$-$\mathtt{Res}$ refutation Π'' of Φ. Since $\mathtt{D}_1$ is also a normal dependency scheme, $\mathtt{LDQ}(\mathtt{D}_1)$-$\mathtt{Res}$ is also sound, and hence, the existence of Π'' implies that Φ is a false QBF.

Completeness: In Theorem 3.16 of [7], $\mathtt{QCDCL}$ (denoted there as $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$) is shown to be complete. Exactly the same proof, which is actually quite intricate, works also to show the completeness of $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$. The idea is as follows: for a false formula Φ, in the 2-player evaluation game, the universal player has a winning strategy on Φ. Since each clause in Φ has a subclause in $\Psi = \mathtt{red}\text{-}\mathtt{D}_1(\Phi)$, the same strategy is also a winning strategy in the evaluation game on Ψ, so Ψ is false. Now, we can construct trails in level order that perform propagations whenever applicable, decide the polarity of existential variables arbitrarily, and decide the polarities of universal variables following this winning strategy. (This is possible because decisions are level-ordered.) The winning strategy guarantees that each such trail runs into a conflict. The set of learnable clauses either contains the empty clause, or is shown to contain an asserting clause – one which after backtracking becomes unit at some point in the trail – and an asserting clause is shown to be new. Thus each trail that does not terminate the refutation learns a new clause, and there are only finitely many clauses that can be added. All the arguments in this outline work also in the presence of a dependency scheme ($\mathtt{D}_2$) that is used in both propagation and learning. ◀

We now wish to examine how adding a particular normal dependency scheme affects the strength of these systems. In this work we focus on adding the *reflexive resolution path* dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ as it is well-studied, and it is known that adding it to $\mathtt{Q}$-$\mathtt{Res}$ gives a strictly stronger system in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-$\mathtt{Res}$. Therefore it is interesting to see if the same parallel extends to the QCDCL systems. Thus in the system $\mathtt{D}_1 + \mathtt{QCDCL}(\mathtt{D}_2)$, we will henceforth assume that $\mathtt{D}_1, \mathtt{D}_2 \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$. When a dependency scheme is $\mathtt{D}^{\mathtt{trv}}$, we will omit reference to it. Thus we have the systems $\mathtt{QCDCL}$, $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$, and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

Before proceeding further, the following propositions are noteworthy to keep in mind.

▶ **Proposition 3.3.** *For a QBF Φ, if $\mathtt{D}(\Phi) = \mathtt{D}^{\mathtt{trv}}(\Phi)$, then all of $\mathtt{QCDCL}$, $\mathtt{QCDCL}(\mathtt{D})$, $\mathtt{D} + \mathtt{QCDCL}$, and $\mathtt{D} + \mathtt{QCDCL}(\mathtt{D})$ are equivalent on Φ and produce the same refutations.*

This is simply because if $\mathtt{D} = \mathtt{D}^{\mathtt{trv}}$, then adding the dependency scheme gives nothing new to the system as no extra reductions are enabled.

▶ **Proposition 3.4.** *For a QBF Φ, $\mathtt{D}^{\mathtt{rrs}}(\Phi) = \emptyset$ if and only if $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is a propositional formula (no universal variables in any clause).*

Further, if $\mathtt{D}^{\mathtt{rrs}}(\Phi) = \emptyset$, then $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ has polynomial size refutations in $\mathtt{Res}$ if and only if Φ has polynomial size refutations in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

Proof (Sketch). If $\mathtt{D}^{\mathtt{rrs}}(\Phi) = \emptyset$, then by definition $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is propositional. If $\mathtt{D}^{\mathtt{rrs}}(\Phi) \neq \emptyset$, then there is some reflexive resolution path involving a universal variable u, and the occurrence of u in the first clause of the path is blocked by an existential literal even with respect to $\mathtt{D}^{\mathtt{rrs}}$. So $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is not propositional.

If $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\Phi)$ is propositional, then in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, after preprocessing, the universal variables have no role to play and the ensuing refutation is a standard $\mathtt{CDCL}$ refutation. Since $\mathtt{CDCL}$ is equivalent to $\mathtt{Res}$, the claim follows. ◀

▶ Remark 3.5. It is tempting to believe that if, for a QBF Φ, $\mathtt{red}\text{-}\mathtt{D}(\Phi)$ is a propositional formula easy to refute in $\mathtt{Res}$, then Φ is easy to refute in $\mathtt{QCDCL}(\mathtt{D})$ as well. However, this intuition is misleading. As we will show in Section 4.5, this is provably not the case.

4 Refutation size bounds for some formulas

In this section we examine the effect of adding the $\mathtt{D}^{\mathtt{rrs}}$ scheme to QCDCL (i.e. the three systems $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$) by computing refutation size bounds for some known QBF formulas, as well as for some newly-constructed QBF formulas.

4.1 The $\mathtt{QParity}_n$ formulas

The first family of formulas that we study are the QParity formulas, first defined in [8].

▶ **Formula 1** ($\mathtt{QParity}_n$). *The* $\mathtt{QParity}_n$ *formula has the prefix* $\exists x_1, \cdots, x_n \forall z \exists t_2, \cdots, t_n$ *and the matrix*

$$\begin{array}{lcccc} & x_1 \vee x_2 \vee \bar{t}_2 & \bar{x}_1 \vee \bar{x}_2 \vee \bar{t}_2 & x_1 \vee \bar{x}_2 \vee t_2 & \bar{x}_1 \vee x_2 \vee t_2 \\ \textit{for } i = 3, \ldots, n: & x_i \vee t_{i-1} \vee \bar{t}_i & x_i \vee \bar{t}_{i-1} \vee t_i & \bar{x}_i \vee t_{i-1} \vee t_i & \bar{x}_i \vee \bar{t}_{i-1} \vee \bar{t}_i \\ & & t_n \vee z & \bar{t}_n \vee \bar{z} & \end{array}$$

As shown in [8], these formulas are hard to refute in QU-Res (and hence also in Q-Res and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$). In [7] it was shown that they have short refutations in QCDCL.

It is straightforward to see that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{QParity}) = \mathtt{D}^{\mathtt{trv}}(\mathtt{QParity})$: the last two clauses give the dependence (z, t_n), and this extends to (z, t_i) for all i using the remaining clauses. Hence the QParity formulas are hard to refute in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$ as well.

On the other hand, since these formulas are easy to refute in QCDCL (and hence also in $\mathtt{QCDCL}^{\mathtt{cube}}$), from Proposition 3.3 it follows that they have short refutations in all three systems: $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$, and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

4.2 The $\mathtt{Equality}_n$ formulas

The family of Equality formulas, introduced in [4], were shown to be hard for QU-Res, and hence also for Q-Res and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$. In [7] they are shown to be hard for QCDCL as well.

▶ **Formula 2** ($\mathtt{Equality}_n$).
The $\mathtt{Equality}_n$ *formula has the prefix* $\exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n$ *and the matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right]$$

In [13], they are shown to be easy to refute in $\mathtt{QCDCL}^{\mathtt{cube}}$. In [3] it is shown that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality}) = \emptyset$, and that the Equality formulas have short refutations in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}$.

$\mathtt{D}^{\mathtt{rrs}} = \emptyset$ also implies $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality})$ is propositional and can be obtained from Equality by dropping all the universals. It is clear to see this formula has a short Res refutation. Therefore by Proposition 3.4 the Equality formulas are easy to refute in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

The formulas are also easy to refute in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, but it is not as straightforward as with the aforementioned case; see Remark 3.5.

▶ **Lemma 4.1.** *The* $\mathtt{Equality}_n$ *formulas have* $O(n^2)$ *refutations in* $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$

Proof (Sketch). For decreasing values of i from $n-1$, construct pairs of trails $\mathcal{U}_i, \mathcal{V}_i$ deciding the variables $x_1 \ldots x_i$ all positively or all negatively, and choose the learnt clause cleverly. ◀

Thus the Equality formulas, which are hard for both Q-Res and QCDCL, become easy to refute when the power of $\mathtt{D}^{\mathtt{rrs}}$ is added to these systems. Thus they showcase the power of dependency schemes and discarding spurious dependencies.

4.3 The $\texttt{Trapdoor}_n$ formulas

The Trapdoor formulas were introduced in [7] to compare QCDCL with $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$. The idea is to juxtapose two propositional formulas, one easy and one hard for Res, and judiciously interleave them with new variables such that QCDCL trails with NO-RED jump to refuting the easy part, while the ones with RED are trapped into refuting the hard part. Thus in QCDCL proof systems enabling more unit propagations is not necessarily an advantage.

▶ **Formula 3** ($\texttt{Trapdoor}_n$). *The* $\texttt{Trapdoor}_n$ *QBF has the prefix* $\exists y_1, \cdots, y_{s_n} \forall w \exists t \exists x_1, \cdots, x_{s_n} \forall u$, *where* s_n *is the number or variables in the propositional pigeonhole principle* $\texttt{PHP}_n^{n+1}$, *and the following matrix:*

$$\begin{array}{lc} & \texttt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \textit{for} \;\; i \in [s_n]: & \bar{y}_i \vee x_i \vee u \;,\; y_i \vee \bar{x}_i \vee u \\ \textit{for} \;\; i \in [s_n]: & y_i \vee w \vee t \;,\; y_i \vee w \vee \bar{t} \;,\; \bar{y}_i \vee w \vee t \;,\; \bar{y}_i \vee w \vee \bar{t} \end{array}$$

Clearly, $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}) = \emptyset$ since the universal variables appear in only one polarity. Thus $\texttt{red-D}^{\texttt{rrs}}(\texttt{Trapdoor})$is obtained by dropping all universals from the Trapdoor formula, and clearly have a short Res refutation using the four y, t clauses for any i. By Proposition 3.4, the Trapdoor formulas are easy to refute in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$. From [7], they are also easy for $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and Q-Res (and hence also $\texttt{Q}(\texttt{D}^{\texttt{rrs}})$-Res and QU-Res). But they are hard in QCDCL because the reductions force unit propagations in the trails which send the solver down a "trap" of refuting PHP.

We observe below that they are easy to refute in $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ as well.

▶ **Lemma 4.2.** *The* $\texttt{Trapdoor}_n$ *formulas have* $O(1)$*-size refutation in* $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$

Proof (Sketch). As seen before, $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}_n) = \emptyset$. Using this fact we can construct a $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ refutation consisting of two trails T_1 and T_2. The first trail decides y_1 and learns $\bar{y}_1$; the second trail has no decisions and learns the empty clause. ◀

4.4 The $\texttt{Dep-Trap}_n$ formulas

The previous sub-sections may suggest that adding $\texttt{D}^{\texttt{rrs}}$ to QCDCL creates a strictly stronger proof system (as is the case with Q-Res.) We design a formula to show that this is not the case. In [7], it is shown that neither of $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and QCDCL simulates the other. This motivates the designing of a formula where adding $\texttt{D}^{\texttt{rrs}}$ enables extra reductions that send the refutation down a "trap" that a seemingly weaker system jumps past. Based on this idea we introduce the Dep-Trap family, a slight modification of the Trapdoor formula.

▶ **Formula 4** ($\texttt{Dep-Trap}_n$). *The* $\texttt{Dep-Trap}_n$ *formula has the prefix* $\exists y_1, \cdots, y_{s_n} \forall w \exists t \forall u \exists x_1, \cdots, x_{s_n}$, *and the matrix is as given below.*

$$\begin{array}{lc} & \texttt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \textit{for} \;\; i \in [s_n]: & \bar{y}_i \vee u \vee x_i \;,\; y_i \vee u \vee \bar{x}_i \\ \textit{for} \;\; i \in [s_n]: & y_i \vee w \vee t \;,\; y_i \vee w \vee \bar{t} \;,\; \bar{y}_i \vee w \vee t \;,\; \bar{y}_i \vee w \vee \bar{t} \\ & \bar{w} \vee \bar{t} \end{array}$$

There are two major differences from the $\texttt{Trapdoor}_n$ formulas. Firstly, u is now quantified before the x_i's; this prevents QCDCL going down the "trap". Secondly, an additional clause $\bar{w} \vee \bar{t}$ is introduced; this prevents the $\texttt{D}^{\texttt{rrs}}$-enabled system from bypassing the "trap".

For exactly the same reasons as Trapdoor, the Dep-Trap formulas are easy to refute in $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and QCDCL, thus also in Q-Res, $\texttt{Q}(\texttt{D}^{\texttt{rrs}})$-Res, QU-Res, and $\texttt{QCDCL}^{\texttt{cube}}$. However these formulas are hard to refute in all QCDCL variants that use $\texttt{D}^{\texttt{rrs}}$.

▶ **Lemma 4.3.** *The* Dep-Trap *formulas have polynomial-size refutations in* QCDCL*,* $\text{QCDCL}^{\text{LEV-ORD}}_{\text{NO-RED}}$*,* Q-Res*,* QU-Res *and* $\text{Q}(\text{D}^{\text{rrs}})$-Res*.*

Proof (Sketch). For QCDCL and $\text{QCDCL}^{\text{LEV-ORD}}_{\text{NO-RED}}$, there is a two-trail refutation, the first deciding all y's positively, then w negatively and learning $(\bar{y}_1)$ from the conflict. The second trail propagates $(\bar{y}_1)$, makes the same decisions to conflict, and learns the empty clause. ◀

▶ **Lemma 4.4.** *Refutations of the* Dep-Trap_n *formulas in* $\text{QCDCL}(\text{D}^{\text{rrs}})$*,* $\text{D}^{\text{rrs}} + \text{QCDCL}$ *and* $\text{D}^{\text{rrs}} + \text{QCDCL}(\text{D}^{\text{rrs}})$ *require exponential size.*

Proof (Sketch). $\text{D}^{\text{rrs}}(\text{Dep-Trap}_n) = \{(w, t)\}$; so w cannot be reduced from axiom clauses during propagation, but u can be reduced. Therefore every y_i decision propagates x_i in the opposite polarity, forcing the trails to refute PHP exactly the same way as hardness of Trapdoor in QCDCL [7]. ◀

The Dep-Trap formulas are thus easy for QCDCL, but become hard to refute when D^{rrs} is added to the system, demonstrating that allowing more reductions and removing spurious dependencies does not necessarily help for the QCDCL system.

4.5 The TwoPHPandCT_n formulas

The previous sections suggest Proposition 3.4 could also hold for $\text{QCDCL}(\text{D}^{\text{rrs}})$, but the following formula refutes that argument. The motivation for this construction also comes from the Trapdoor formulas, with the new element of using two copies of the hard part.

▶ **Formula 5** (TwoPHPandCT_n)**.** *The* TwoPHPandCT_n *formulas has the prefix* $\mathcal{Q} = \forall u \exists x_1 \cdots x_{s_n}\ \exists y_1 \cdots y_{s_n}\ \forall v \exists z_1, z_2$ *and the matrix*

$$u \vee \text{PHP}(x_1, \cdots, x_{s_n}) \qquad \bar{u} \vee \text{PHP}(y_1, \cdots, y_{s_n})$$
$$v \vee z_1 \vee z_2\ ,\ v \vee \bar{z}_1 \vee z_2\ ,\ v \vee z_1 \vee \bar{z}_2\ ,\ v \vee \bar{z}_1 \vee \bar{z}_2$$

Observe that these formulas are easy to refute in Q-Res, using the four z_1, z_2 clauses, and hence also easy to refute in $\text{Q}(\text{D}^{\text{rrs}})$-Res and QU-Res.

Since v appears in only one polarity and $u, \bar{u}$ appear in clauses with disjoint variables, hence $\text{D}^{\text{rrs}}(\text{TwoPHPandCT}) = \emptyset$ and $\text{red-D}^{\text{rrs}}(\text{TwoPHPandCT})$ is the propositional formula of two copies of PHP and four clauses of the complete tautology on two variables, and hence is easy to refute in Res. By Proposition 3.4, the original QBFs are easy to refute in $\text{D}^{\text{rrs}} + \text{QCDCL}$ and $\text{D}^{\text{rrs}} + \text{QCDCL}(\text{D}^{\text{rrs}})$.

However they are hard for all the three systems QCDCL, $\text{QCDCL}(\text{D}^{\text{rrs}})$ and $\text{QCDCL}^{\text{LEV-ORD}}_{\text{NO-RED}}$.

▶ **Lemma 4.5.** *The QBF formulas* TwoPHPandCT_n *require exponential size refutations in* QCDCL*,* $\text{QCDCL}(\text{D}^{\text{rrs}})$ *and* $\text{QCDCL}^{\text{LEV-ORD}}_{\text{NO-RED}}$*.*

Proof (Sketch). All three systems have no preprocessing and must first decide u, which causes no propagations, and then are reduced to refuting PHP (in either x or y), which requires exponential size refutations. ◀

These formulas highlight two important facts: firstly that $\text{QCDCL}(\text{D}^{\text{rrs}})$ is not the same as $\text{D}^{\text{rrs}} + \text{QCDCL}$ or $\text{D}^{\text{rrs}} + \text{QCDCL}(\text{D}^{\text{rrs}})$ and does not simulate them either. And secondly, even in the case when $\text{D}^{\text{rrs}} = \emptyset$ and reducing the formula by D^{rrs} gives us an easy propositional formula, it can still be hard to refute for $\text{QCDCL}(\text{D}^{\text{rrs}})$.

4.6 The $\texttt{RRSTrapEq}_n$ formulas

The next family of formulas are obtained by making a slight modification to the Equality formulas. The motivation to define such formulas comes from trying to ascertain whether after preprocessing with $\texttt{D}^{\texttt{rrs}}$, does allowing reductions using $\texttt{D}^{\texttt{rrs}}$ for unit propagation add any power over trivial universal reductions.

▶ **Formula 6** ($\texttt{RRSTrapEq}_n$). *The* $\texttt{RRSTrapEq}_n$ *formula has the prefix* $\exists a \exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n \exists b$ *and the PCNF matrix as below:*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i \vee b)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i \vee b)}_{B_i} \right] \wedge \bigwedge_{i=1}^{n} \underbrace{(u_i \vee \bar{b})}_{C_i} \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

The underlying idea in the formulation is that unlike $\texttt{D}^{\texttt{rrs}}(\texttt{Equality})$ which is empty, we add the C_i clauses to make $\texttt{D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \{(u_i, b) : i \in [n]\}$, and add b to the x, u, t clauses to obtain $\texttt{red-D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \texttt{RRSTrapEq}$. Consequently, preprocessing by $\texttt{D}^{\texttt{rrs}}$ does not change the formula at all. Therefore the refutation for these formulas will be exactly the same for QCDCL and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$, and similarly for $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$.

▶ **Lemma 4.6.** *The* RRSTrapEq *formulas have polynomial size refutations in* $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ *and* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$*, but require exponential size refutations in* QCDCL *and* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}$

Proof (Sketch). Since, $\texttt{red-D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \texttt{RRSTrapEq}$, preprocessing has no effect. After the first decision, hardness reduces to hardness of the Equality formulas (Section 4.2) in the QCDCL or $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$systems. ◀

Additionally, since the Equality formulas are embedded in the RRSTrapEq formula, and since QU-Res is closed under restrictions, the RRSTrapEq formulas are hard for QU-Res and in turn Q-Res and $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$. However, they have easily seen short refutations in $\texttt{Q}(\texttt{D}^{\texttt{rrs}})\texttt{-Res}$.

4.7 The $\texttt{PreDepTrap}_n$ formulas

The previous section underlines that preprocessing using $\texttt{D}^{\texttt{rrs}}$ does not make $\texttt{D}^{\texttt{rrs}}$ during propagation obsolete. However, one might expect that at the very least preprocessing won't make things worse. However that is not so; the following example highlights that fact.

▶ **Formula 7** ($\texttt{PreDepTrap}_n$). *The* $\texttt{PreDepTrap}_n$ *formula has the prefix* $\forall a\ \exists y_1 \cdots y_{s_n} \forall w \exists t \forall u\ \exists x_1 \cdots x_{s_n}\ \exists p_1 \cdots p_n \forall q_1 \cdots q_n \exists r_1 \cdots r_n$*, and the matrix*

$$a \vee \texttt{Dep-Trap}(y_1, \cdots, y_{s_n}, w, t, u, x_1, \cdots, x_{s_n})$$
$$\bar{a} \vee \texttt{Equality}(p_1, \cdots, p_n, q_1, \cdots, q_n, r_1, \cdots, r_n)$$

Looking at $\texttt{D}^{\texttt{rrs}}$ for the formula, the literals a and $\bar{a}$ appear in clauses with disjoint non-interacting sets of variables, and $\texttt{D}^{\texttt{rrs}}(\texttt{Equality}) = \emptyset$. So $\texttt{D}^{\texttt{rrs}}(\texttt{PreDepTrap}) = \texttt{D}^{\texttt{rrs}}(\texttt{Dep-Trap})$

Consider the systems $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$, QCDCL and $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$. Every trail must start with setting the variable a. In the latter system, let the trails start with decision a, which reduces the formula to exactly the Equality formulas, which are easy to refute in $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$, (Lemma 4.1). Thus the same refutation as Lemma 4.1 with a leading decision a in every trail gives an easy $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ refutation. In the two former systems, let the trails decide $\bar{a}$. Then the formula reduces exactly to Dep-Trap formula which is easy to refute in $\texttt{QCDCL}^{\texttt{LEV-ORD}}_{\texttt{NO-RED}}$ and QCDCL (Section 4.4) and as a consequence in Q-Res, $\texttt{Q}(\texttt{D}^{\texttt{rrs}})\texttt{-Res}$, QU-Res, and $\texttt{QCDCL}^{\texttt{cube}}$.

Finally, we show that preprocessing the formulas makes refutations exponentially long.

▶ **Lemma 4.7.** *The* PreDepTrap *formulas require exponential size refutations in* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ *and* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$

Proof (Sketch). As in the case of Trapdoor, QCDCL trails on $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\mathtt{PreDepTrap})$ are forced to refute the PHP part. ◀

4.8 The $\mathtt{PropDep\text{-}Trap}_n$ formulas

Section 4.7 illustrates that $\mathtt{D}^{\mathtt{rrs}}$ as a preprocessing technique was detrimental; it was better to use $\mathtt{D}^{\mathtt{rrs}}$ *only* in unit propagation. This begs the question – could there be a formula such that $\mathtt{D}^{\mathtt{rrs}}$ only as preprocessing is strictly better than $\mathtt{D}^{\mathtt{rrs}}$ for just propagation or both preprocessing and propagation? The answer is yes, and is witnessed by the following slight modification of the Dep-Trap formulas

▶ **Formula 8** ($\mathtt{PropDep\text{-}Trap}_n$)**.** *The* $\mathtt{PropDep\text{-}Trap}_n$ *formulas have the prefix* $\exists s \;\; \exists y_1 \cdots y_{s_n} \forall w \exists t \forall b_1, b_2 \; \exists x_1 \cdots x_{s_n} \; \exists z_1, z_2$ *and the matrix as given below.*

$$\begin{array}{ll} & \mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \textit{for} \;\; i \in [s_n]: & \bar{y}_i \vee b_1 \vee x_i \vee z_1 \;,\; y_i \vee b_2 \vee \bar{x}_i \vee z_2 \\ & s \vee w \vee t \;,\; s \vee w \vee \bar{t} \;,\; \bar{s} \vee w \vee t \;,\; \bar{s} \vee w \vee \bar{t} \;, \quad \bar{w} \vee \bar{t} \;, \\ & \bar{b}_1 \vee \bar{z}_1 \;,\; \bar{b}_2 \vee \bar{z}_2 \;,\; \bar{z}_1 \;,\; \bar{z}_2 \end{array}$$

Notice that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap}) = \{(w,t),(b_1,z_1),(b_2,z_2)\}$. Therefore preprocessing has no effect on the formula; $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap}) = \mathtt{PropDep\text{-}Trap}$.

The presence of the four s, w, t clauses make this formula very easy to refute in Q-Res and hence also in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res and QU-Res. They are also easy to refute in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, QCDCL, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{QCDCL}^{\mathtt{cube}}$ because, apart from the initial propagation of $\bar{z}_1, \bar{z}_2$, the s,y_i decisions cause no propagation, then we can decide $\bar{w}$ and reach a conflict using the s, w, t clauses. If the trail decided $s(\bar{s})$, from this conflict we can learn $\bar{s}(s)$; the next trail propagtes $\bar{s}(s)$, and after deciding $\bar{w}$ in the same manner the empty clause can be learnt.

These formulas are however hard to refute in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

▶ **Lemma 4.8.** *The* PropDep-Trap *formulas require exponential size refutations in* $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ *and* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$

Proof (Sketch). $\mathtt{red}\text{-}\mathtt{D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap}) = \mathtt{PropDep\text{-}Trap}$; therefore preprocessing has no effect. $\mathtt{D}^{\mathtt{rrs}}$ during propagation causes propagation of x_i due to y_i forcing the refutation down the same "trap" as the Trapdoor formulas for QCDCL[7]. ◀

4.9 The $\mathtt{TwinEq}_n$ formulas

The TwinEq formulas were introduced in [13] to show hardness in the system $\mathtt{QCDCL}^{\mathtt{cube}}$. They are formally defined as follows:

▶ **Formula 9** ($\mathtt{TwinEq}_n$ [13])**.** *The* $\mathtt{TwinEq}_n$ *formula has the prefix* $\exists x_1 \cdots x_n \forall u_1 \cdots u_n, w_1, \cdots, w_n \exists t_1 \cdots t_n$ *and the PCNF matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right] \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee w_i \vee t_i)}_{C_i} \wedge \underbrace{(\bar{x}_i \vee \bar{w}_i \vee t_i)}_{D_i} \right]$$

These formulas are hard for $\mathtt{QCDCL}^{\mathtt{cube}}$, and hence also for $\mathtt{QCDCL}$ and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$. For the same reason as for $\mathtt{Equality}$ (the size-cost-capacity theorem from [4]), they are also hard for $\mathtt{QU\text{-}Res}$ and $\mathtt{Q\text{-}Res}$.

It is easy to show that $\mathtt{D}^{\mathtt{rrs}}(\mathtt{TwinEq}) = \emptyset$ (just as $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality})$ is shown to be $\emptyset$). Therefore, $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{TwinEq})$ is a propositional formula; in fact is the same formula as $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{Equality})$, which has a short Resolution proof. Hence, $\mathtt{TwinEq}$ is easy to refute in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\mathtt{\text{-}Res}$, and from Proposition 3.4, is also easy to refute in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$, $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$. They are also easy to refute in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, following the same argument as for the $\mathtt{Equality}$ formulas.

5 Relations between proof systems

Putting together the bounds from the previous section, we can now place the newly-defined proof systems within the simulation order. First we observe that the four versions of $\mathtt{QCDCL}$ that use or do not use $\mathtt{D}^{\mathtt{rrs}}$ in either of the two ways are all pairwise incomparable. Next we observe that each of the three new versions of $\mathtt{QCDCL}$ is also incomparable with $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, $\mathtt{Q\text{-}Res}$, $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\mathtt{\text{-}Res}$, and $\mathtt{QU\text{-}Res}$. Finally, we observe that even when we add cube-learning to standard $\mathtt{QCDCL}$, the system $\mathtt{QCDCL}^{\mathtt{cube}}$ is still incomparable with all the three versions of $\mathtt{QCDCL}$ with dependency scheme added.

▶ **Theorem 5.1.** *The proof systems in* $\{\mathtt{QCDCL}, \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}}), \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}, \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})\}$ *are pairwise incomparable.*

Proof. Of the four systems under consideration, the $\mathtt{Trapdoor}$ formulas (Section 4.3) are hard only for $\mathtt{QCDCL}$, and the $\mathtt{Dep\text{-}Trap}$ formulas (Section 4.4)are easy only in $\mathtt{QCDCL}$. Hence $\mathtt{QCDCL}$ is incomparable with all three systems obtained by adding $\mathtt{D}^{\mathtt{rrs}}$.

Among the three systems using $\mathtt{D}^{\mathtt{rrs}}$, the $\mathtt{TwoPHPandCT}$ formulas (Section 4.5) are hard only in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, while the $\mathtt{PreDepTrap}$ formulas (Section 4.7) are easy only in $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$. Hence $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ is incomparable with the systems that use preprocessing.

Finally, the systems $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ are separated by the formulas $\mathtt{RRSTrapEq}$ (Section 4.6) easy only in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, and the formulas $\mathtt{PropDep\text{-}Trap}$ (Section 4.8) easy only in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$. ◀

▶ **Theorem 5.2.** *Any two proof systems* $\mathtt{P}_1 \in \{\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}}), \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}, \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})\}$, *and* $\mathtt{P}_2 \in \{\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}, \mathtt{Q\text{-}Res}, \mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\mathtt{\text{-}Res}, \mathtt{QU\text{-}Res}\}$, *are incomparable.*

Proof. The $\mathtt{QParity}$ formulas (Section 4.1) require exponential size refutations in $\mathtt{P}_2$ but have polynomial size refutations in $\mathtt{P}_1$.

The $\mathtt{Dep\text{-}Trap}$ formulas (Section 4.4) have constant size refutations in $\mathtt{P}_2$ but require exponential size refutations in $\mathtt{P}_1$. ◀

▶ **Theorem 5.3.** *Every proof system in* $\{\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}}), \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}, \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})\}$ *is incomparable with* $\mathtt{QCDCL}^{\mathtt{cube}}$.

Proof. The $\mathtt{TwinEq}$ formulas (Section 4.9) require exponential size refutations in $\mathtt{QCDCL}^{\mathtt{cube}}$ but have poly-size refutations in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$, $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$.

The $\mathtt{Dep\text{-}Trap}$ formulas (Section 4.4) require exponential size refutations in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}$, $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$, but have short refutations in $\mathtt{QCDCL}^{\mathtt{cube}}$. ◀

6 Conclusion

We have examined, from a rigourous proof-theoretic viewpoint, the effect of incorporating heuristics based on dependency schemes into QBF solving algorithms based on the conflict-driven clause learning paradigm. Our results show that unlike in the case of the proof system Q-Res, where dependency-awareness can shorten refutations but never lengthens them, here the picture is much more nuanced, and all kinds of shortenings as well as lengthenings can be observed. Thus the decision of whether or not to make a QCDCL solver account for spurious dependencies is itself a challenging one, and it is likely the domain from where instances are to be solved may indicate what choice is more suitable.

Some directions for further studies include

1. the impact of other dependency schemes; e.g. the less (than $\mathtt{D}^{\mathtt{rrs}}$) general standard dependency scheme $\mathtt{D}^{\mathtt{std}}$ introduced in [21], and more general schemes based on tautology-free and implication-free paths introduced in [5, 6]. It seems reasonable to expect that a similar nuanced picture will present when considering such schemes as well.
2. the impact of specific learning schemes on dependency-aware QCDCL. Our lower bounds hold for any QCDCL-based solver as long as the learning scheme picks a clause only from the learnable-clause sequence as defined in Section 2. However, the upper bounds hold for specific choices of learnt clauses, and this, in some sense, reflects a certain non-determinism in the algorithm. (This is somewhat akin to the non-determinism inherent in CDCL algorithms in the statement that CDCL simulates Resolution.) Arguably, the upper bounds will be more meaningful if achieved with actually-used learning schemes. While some of our upper bounds are achieved with such schemes, specifically the UIP policy, some others make ad hoc choices with respect to which clauses to learn.
3. formalising the dependency-learning approach (used in the solver Qute [18]). Here the solver adds detected dependencies starting from $\emptyset$, rather than removing spurious dependencies starting from $\mathtt{D}^{\mathtt{trv}}$. This would change the decision order from LEV-ORD. Recent work in [11] shows that other orders are not necessarily unsound, but we are quite short on techniques to analyse any-decision-order.
4. formalising the combination of cube-learning and dependency schemes for all true and false QBFs. This may be somewhat non-trivial and nuanced, as adding dependency schemes to the formal proof system of long-distance term resolution (the proof system in which proofs can be extracted from runs of solvers on true QBFs, Q-consensus) is not known to be sound (see the Discussion section in [19]).

References

1 Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res.*, 40:353–373, 2011. doi:10.1613/jair.3152.

2 Salman Azhar, Gary Peterson, and John Reif. Lower bounds for multiplayer non-cooperative games of incomplete information. *Journal of Computers and Mathematics with Applications*, 41:957–992, 2001.

3 Olaf Beyersdorff and Joshua Blinkhorn. Dynamic QBF dependencies in reduction and expansion. *ACM Trans. Comput. Log.*, 21(2):8:1–8:27, 2020. doi:10.1145/3355995.

4 Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random QBFs. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:13)2019.

5 Olaf Beyersdorff, Joshua Blinkhorn, and Tomás Peitl. Strong (D)QBF dependency schemes via tautology-free resolution paths. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing – SAT 2020 – 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 394–411. Springer, 2020. doi:10.1007/978-3-030-51825-7_28.

6 Olaf Beyersdorff, Joshua Blinkhorn, and Tomás Peitl. Strong (D)QBF dependency schemes via implication-free resolution paths. *Electron. Colloquium Comput. Complex.*, TR21-135, 2021. arXiv:TR21-135.

7 Olaf Beyersdorff and Benjamin Böhm. Understanding the Relative Strength of QBF CDCL Solvers and QBF Resolution. *Logical Methods in Computer Science*, Volume 19, Issue 2, April 2023. (Preliminary version in ITCS'21). doi:10.46298/lmcs-19(2:2)2023.

8 Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. New resolution-based QBF calculi and their proof complexity. *ACM Trans. Comput. Theory*, 11(4):26:1–26:42, 2019. doi:10.1145/3352155.

9 Olaf Beyersdorff, Mikolás Janota, Florian Lonsing, and Martina Seidl. Quantified boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability – Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1177–1221. IOS Press, 2021. doi:10.3233/FAIA201015.

10 Joshua Blinkhorn and Olaf Beyersdorff. Shortening QBF proofs with dependency schemes. In *Theory and Applications of Satisfiability Testing – SAT*, volume 10491 of *LNCS*, pages 263–280. Springer, 2017. doi:10.1007/978-3-319-66263-3_17.

11 Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. Should decisions in QCDCL follow prefix order? In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 11:1–11:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SAT.2022.11.

12 Benjamin Böhm and Olaf Beyersdorff. QCDCL vs QBF resolution: Further insights. In *26th Theory and Applications of Satisfiability Testing – SAT*, 2023. full version in ECCC, TR23-051.

13 Benjamin Böhm, Tomáš Peitl, and Olaf Beyersdorff. QCDCL with cube learning or pure literal elimination – What is best? In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1781–1787. International Joint Conferences on Artificial Intelligence Organization, July 2022. Main Track. doi:10.24963/ijcai.2022/248.

14 Abhimanyu Choudhury and Meena Mahajan. Dependency schemes in CDCL-based QBF solving: a proof-theoretic study. *Electron. Colloquium Comput. Complex.*, TR23-061, 2023. arXiv:TR23-061.

15 Florian Lonsing. *Dependency schemes and search-based QBF solving: theory and practice*. PhD thesis, Johannes Kepler University, Linz, Austria, 2012.

16 Florian Lonsing and Armin Biere. Depqbf: A dependency-aware QBF solver. *J. Satisf. Boolean Model. Comput.*, 7(2-3):71–76, 2010. doi:10.3233/sat190077.

17 Florian Lonsing and Armin Biere. Integrating dependency schemes in search-based QBF solvers. In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing – SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2010. doi:10.1007/978-3-642-14186-7_14.

18 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency learning for QBF. *J. Artif. Intell. Res.*, 65:180–208, 2019. doi:10.1613/jair.1.11529.

19 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-distance Q-resolution with dependency schemes. *J. Autom. Reasoning*, 63(1):127–155, 2019. doi:10.1007/s10817-018-9467-3.

20 Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.*, 175(2):512–525, 2011. doi:10.1016/j.artint.2010.10.002.

21 Marko Samer and Stefan Szeider. Backdoor sets of quantified boolean formulas. *J. Autom. Reason.*, 42(1):77–97, 2009. doi:10.1007/s10817-008-9114-5.

22 Christoph Scholl and Ralf Wimmer. Dependency quantified Boolean formulas: An overview of solution methods and applications – extended abstract. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *International Conference on Theory and Practice of Satisfiability Testing SAT*, volume 10929 of *LNCS*, pages 3–16. Springer, 2018.

23 Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified boolean formulas. In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*, pages 78–84. IEEE, 2019. doi:10.1109/ICTAI.2019.00020.

24 Friedrich Slivovsky and Stefan Szeider. Soundness of Q-resolution with dependency schemes. *Theor. Comput. Sci.*, 612:83–101, 2016. doi:10.1016/j.tcs.2015.10.020.

Weighted One-Deterministic-Counter Automata

Prince Mathew
Indian Institute of Technology Goa, India

Vincent Penelle
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France

Prakash Saivasan
The Institute of Mathematical Sciences, HBNI, India
CNRS UMI ReLaX, India

A.V. Sreejith
Indian Institute of Technology Goa, India

Abstract

We introduce weighted one-deterministic-counter automata (ODCA). These are weighted one-counter automata (OCA) with the property of counter-determinacy, meaning that all paths labelled by a given word starting from the initial configuration have the same counter-effect. Weighted ODCAs are a strict extension of weighted visibly OCAs, which are weighted OCAs where the input alphabet determines the actions on the counter.

We present a novel problem called the co-VS (complement to a vector space) reachability problem for weighted ODCAs over fields, which seeks to determine if there exists a run from a given configuration of a weighted ODCA to another configuration whose weight vector lies outside a given vector space. We establish two significant properties of witnesses for co-VS reachability: they satisfy a pseudo-pumping lemma, and the lexicographically minimal witness has a special form. It follows that the co-VS reachability problem is in P.

These reachability problems help us to show that the equivalence problem of weighted ODCAs over fields is in P by adapting the equivalence proof of deterministic real-time OCAs [3] by Böhm et al. This is a step towards resolving the open question of the equivalence problem of weighted OCAs. Finally, we demonstrate that the regularity problem, the problem of checking whether an input weighted ODCA over a field is equivalent to some weighted automaton, is in P. We also consider boolean ODCAs and show that the equivalence problem for (non-deterministic) boolean ODCAs is in PSPACE, whereas it is undecidable for (non-deterministic) boolean OCAs.

2012 ACM Subject Classification Theory of computation → Automata extensions

Keywords and phrases One-counter automata, Equivalence, Weighted automata, Reachability

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.39

Related Version *Full Version*: https://arxiv.org/abs/2301.13456

Funding *A.V. Sreejith*: supported by SERB for the project "Probabilistic Pushdown Automata" [MTR/2021/000788].

Acknowledgements The authors would like to thank Rahul C S for his intuitive suggestions that helped in proving Lemma 14.

1 Introduction

This paper investigates a restriction on weighted one-counter automata (OCA). Like weighted finite automata, weighted OCAs recognise functions - every word over a finite alphabet is mapped to a weight. We say that a weighted OCA has *counter-determinacy* (see Definition 6) if "all paths labelled by a given word, starting from the initial configuration, have the same counter-effect". Weighted one-deterministic-counter automata (ODCA) is a syntactic model equivalent to weighted OCA with counter-determinacy (see Definition 7). It consists of,

© Prince Mathew, Vincent Penelle, Prakash Saivasan, and A.V. Sreejith;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 39; pp. 39:1–39:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1. Counter: A counter that stays non-negative and allows zero tests.
2. Counter structure: A finite state deterministic machine where the transitions depend only on its current state, the input letter, and whether the counter is zero. The counter structure can increment/decrement the counter by one, or leave it unchanged.
3. Finite state machine: A finite state weighted automaton whose transitions depend on its current state, the input letter, and whether the counter value is zero. This machine cannot modify the counter.

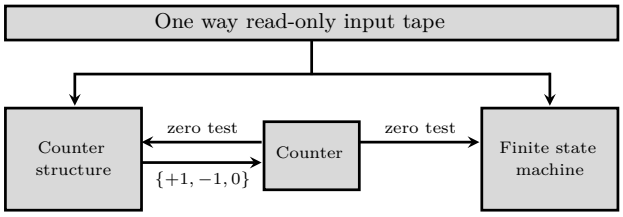


Figure 1 One-deterministic-counter automata.

The counter structure and the finite state machine run synchronously on any word. The finite state machine computes the weight associated with the word. Our first observation is:

▶ **Theorem 1.** *There is a polynomial time translation from a weighted* OCA *with counter-determinacy to a weighted* ODCA *and vice versa.*

In the following example, the functions `prefixAwareDecimal` and `equalPrefixPower` are recognised by weighted OCA with counter-determinacy.

▶ **Example 2.** *The functions are defined over the alphabet* $\Sigma = \{a, b\}$*. The transition weights of the* ODCA*s are from the field of rational numbers* $\mathbb{Q}$*.*

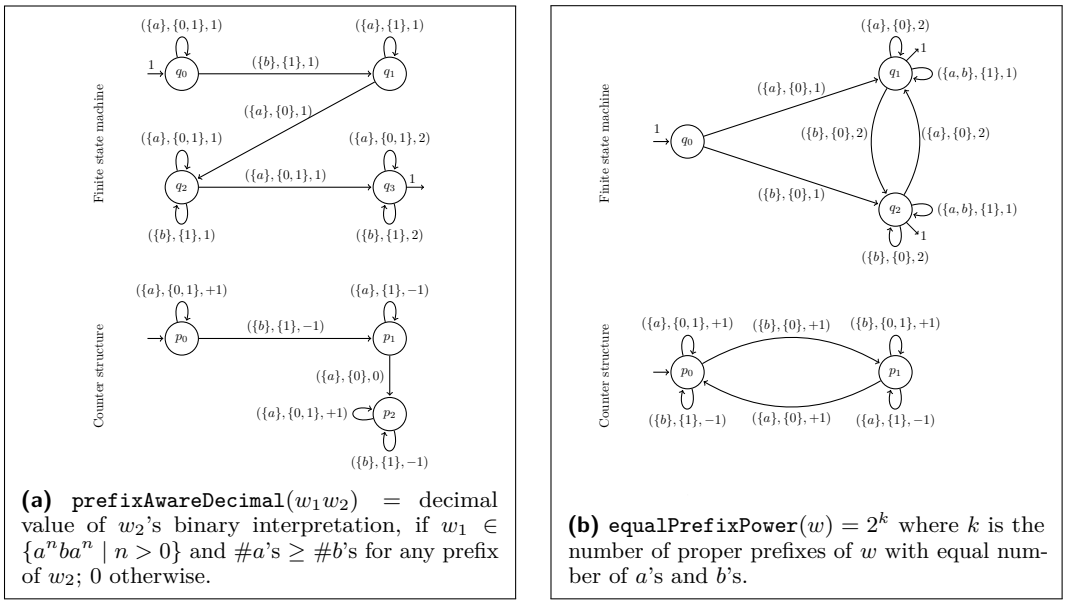
(a) *The function* `prefixAwareDecimal` $: \Sigma^* \to \mathbb{N}$ *is defined as follows:* `prefixAwareDecimal`$(w) =$ `decimal`(w_2) *if* $w = w_1 w_2$*,* $w_1 \in \{a^n b a^n \mid n > 0\}$*, and the number of* a*'s* $\geq$ *number of* b*'s for any prefix of* w_2*, and* 0 *otherwise. Here,* `decimal`(w_2) *represents the decimal equivalent of* w_2 *when interpreted as a binary number, where "a" is treated as a one and "b" as a zero.*

(b) *The function* `equalPrefixPower` $: \Sigma^* \to \mathbb{N}$ *is defined as follows: for all* $w \in \Sigma^*$*,* `equalPrefixPower`$(w) = 2^k$ *where* k *is the number of proper prefixes of* w *with equal number of* a*'s and* b*'s.*

The weighted ODCAs recognising these functions are given in Figure 2. In the figure, if a transition from p_i to p_j of the *counter structure* is labelled (A, R, D) and $(a, r, d) \in A \times R \times D$, then there is a transition from p_i to p_j on reading the symbol a with counter action d. If a transition from q_i to q_j of the *finite state machine* is labelled (A, R, s) and $(a, r, s) \in A \times R \times \mathbb{Q}$, then there is a transition from q_i to q_j on reading the symbol a with weight s. In both cases, r denotes the sign of the counter value. i.e., the current counter value should be 0 if $r = 0$ and greater than 0 if $r = 1$. For the finite state machine, the initial (resp. output) weight is marked using an inward (resp. outward) arrow. The weight of a path is the product of transition weights along that path. The accepting weight of a word is the sum of weights of all the paths from an initial state to an output state labelled by that word.

1.1 Comparisons with other models

Visibly pushdown automata (VPDA) were introduced by Alur and Madhusudan in 2004 [2]. They have received much attention as they are a strict subclass of pushdown automata suitable for program analysis. VPDAs enjoy tractable decidable properties, which are undecidable



(a) `prefixAwareDecimal`(w_1w_2) = decimal value of w_2's binary interpretation, if $w_1 \in \{a^nba^n \mid n > 0\}$ and $\#a$'s $\geq \#b$'s for any prefix of w_2; 0 otherwise.

(b) `equalPrefixPower`$(w) = 2^k$ where k is the number of proper prefixes of w with equal number of a's and b's.

Figure 2 The figure shows weighted ODCAs recognising the functions given in Example 2.

in the general case. The visibly restriction, in essence, is that the stack operations are *input-driven*, i.e., only depends on the letter read. Weighted VPDA is a natural extension to the weighted setting. Counter-determinacy can be seen as a relaxation in the visibly constraint on OCAs, as the counter actions are no longer input-driven but are deterministic. The fact that weighted ODCAs are strictly more expressive than weighted visibly OCA can be noted from the fact that the functions in Example 2 are not recognised by a weighted visibly OCA.

Nowotka et al. [17] introduced height-deterministic pushdown automata, where the input string determines the stack height. Weighted ODCAs can be seen as weighted height-deterministic pushdown automata over a single stack alphabet and a bottom-of-stack symbol.

The reader might feel that a weighted ODCA is equivalent to a cartesian product of a deterministic OCA and a weighted finite automaton. However, one can note that the functions `prefixAwareDecimal` and `equalPrefixPower` in Example 2 are not definable by the cartesian product of deterministic OCA and a weighted automaton. The reason is that the weighted automaton cannot "see" the counter values, so its power is restricted.

1.2 Motivation

Probabilistic pushdown automata (PPDA) have been studied for the analysis of stochastic programs with recursion [14, 19]. They are equivalent to recursive Markov chains [8, 15]. PPDAs are also a generalisation of stochastic context-free grammars [1] used in natural language processing and many variants of one-dimensional random walks [7].

The decidability of equivalence of probabilistic pushdown automata is a long-standing open problem [10]. The problem is inter-reducible to multiplicity equivalence of context-free grammars. In fact, the decidability is only known for some special subclasses of PPDA. It is known that the equivalence problem for PPDA is in PSPACE if the alphabet contains only one letter and is at least as hard as polynomial identity testing [10]. There is a randomised polynomial time algorithm that determines the non-equivalence of two visibly PPDA over the

alphabet triple $(\Sigma_{call}, \Sigma_{ret}, \Sigma_{int})$ where both machines perform push, pop, and no-action on the stack over the symbols in Σ_{call}, Σ_{ret}, and Σ_{int} respectively [12]. There is a polynomial-time reduction from polynomial identity testing to this problem. Hence it is highly unlikely that the problem is in P.

Since the equivalence problem for pPDA is unknown, the natural question to ask is the equivalence problem for probabilistic one-counter automata. However, this problem is also unresolved. In this paper, we identify a subclass of probabilistic OCAs (probabilistic ODCAs are also a superclass of visibly probabilistic OCAs) for which the equivalence problem is decidable. In particular, we show that the problem is in P. Note that our results are slightly more general since we consider weighted ODCAs where weights are from a field.

1.3 Our contributions on weighted ODCA (weights from a field)

The paper's primary focus is on the equivalence problem for weighted ODCAs where the weights are from a field (possibly infinite).

We first introduce a novel reachability problem on weighted ODCA, called the complement to a vector space (co-VS) reachability problem. The co-VS reachability problem (see Section 3) takes a weighted ODCA, an initial configuration, a vector space, a final counter state, and a final counter value as input. It asks, starting from the initial configuration, whether it is possible to reach a configuration with the final counter state, final counter value, and weight distribution over the states that is not in the vector space.

Let us call a word a *witness* if the run of the word "reaches" a configuration desired by the reachability problem. We identify two interesting properties of witnesses.

1. pseudo-pumping lemma (Lemma 10): If the run of a witness encounters a "large" counter value, then it can be pumped-down (resp. pumped-up) to get a run where the maximum counter value encountered is smaller (resp. larger). However, the lemma is distinct from a traditional pumping lemma, where the same subword can be pumped-down (or pumped-up) multiple times while maintaining reachability. In the case of a weighted ODCA, we only claim that a subword can be pumped, but the same subword may not be repeatedly pumped. It follows from the pseudo-pumping lemma that the co-VS reachability problem is in P (Theorem 13).
2. special-word lemma (Lemma 15): The lexicographically smallest witness is of the form $uy_1^{r_1}vy_2^{r_2}w$ where u, v, w, y_1 and y_2 are "small" words and $r_1, r_2 \in \mathbb{N}$. The length of the word uy_1vy_2w is bounded by a polynomial in the number of states of the ODCA, whereas r_1 and r_2 also depend on the counter values of the initial and final configurations.

Comparing the above properties with that of deterministic one-counter automata will be interesting. In a deterministic OCA, the reachability problem is equivalent to asking whether there is a path to a final state (rather than a weight distribution over states) and a counter value from an initial state and counter value. Let z be an arbitrary "long" witness. Consider the run on z of the deterministic OCA. By the Pigeonhole principle (see Valiant and Paterson [24]), there will be words u, y_1, v, y_2, and w such that $z = uy_1vy_2w$, and y_1 (and similarly y_2) starts and ends in the same state and the effect of y_1 on the counter is minus of the effect of y_2 on the counter. In short, y_1 and y_2 form loops with inverse counter-effects and can be pumped simultaneously. Therefore, for all $r \in \mathbb{N}$, the word $uy_1^rvy_2^rw$ is a witness. One can view this as a pumping lemma for OCA (see Ogden's lemma [18] for pushdown automata). Such a property does not hold in the case of weighted ODCA. The presence of weights at each state makes the problem inherently complex, necessitating a more sophisticated approach.

The proofs of Lemma 10 and Lemma 15 use linear algebra and combinatorics on words and are distinct from those employed for deterministic OCA. We also introduce a similar problem called co-VS coverability (see Section 3). The two properties of the witness and co-VS coverability are crucial along with the ideas developed by Böhm et al. [3, 4, 6] and Valiant and Paterson [24] in solving the equivalence problem. The proof is rather technical, Section 4 provides a high-level idea of the proof.

▶ **Theorem 3.** *There is a polynomial time algorithm that decides if two weighted ODCAs (weights from a field) are equivalent and outputs a word that distinguishes them otherwise.*

Finally, we consider the regularity problem - the problem of deciding whether a weighted ODCA is equivalent to some weighted automaton. The proof technique is adapted from the ideas developed by Böhm et al. [6] in the context of real-time OCA. The crucial idea in proving regularity is to check for the existence of infinitely many equivalence classes. The pseudo-pumping lemma (particularly pumping-up) is used in proving this. A detailed proof can be found in Appendix B.

▶ **Theorem 4.** *The regularity problem of weighted ODCA (weights from a field) is in* P.

1.4 Related work

Extensive studies have been conducted on weighted automata with weights from semirings. Tzeng [23] (also see Schützenberger [20]) gave a polynomial time algorithm to decide the equivalence of two probabilistic automata. The result has been extended to weighted automata with weights over a field. On the other hand, the problem is undecidable if the weights are over the semiring $(\mathbb{N}, \min, +)$ [13]. Unlike the extensive literature on weighted automata, the study on weighted versions of pushdown or one-counter machines is limited [9, 11, 14]. The undecidability of several interesting problems creates a major bottleneck.

Moving on to the non-weighted models, the equivalence problem for non-deterministic pushdown automata is known to be undecidable. On the other hand, from the seminal result by Sénizergues [21], we know that the equivalence problem for deterministic pushdown automata is decidable. It was later proved to be primitive recursive [22]. The language equivalence of synchronised real-time height-deterministic pushdown automata is in EXPTIME [17]. The equivalence problem for deterministic one-counter automata (with and without ϵ transitions), similar to that of deterministic finite automata, is NL-complete [5].

2 Preliminaries

2.1 Basic notations

In this paper, we fix an alphabet Σ. Given a word $w \in \Sigma^*$, we use $|w|$ to denote the length of the word w. For any set S, we use $|S|$ to denote the number of elements in S. We use the notation $[i, j]$ to denote the interval $\{i, i+1, \ldots, j\}$. We say that a word $u = a_1 \cdots a_k$ is a subword of a word w, if $w = u_0 a_1 u_1 a_2 \cdots a_k u_k$, where $a_i \in \Sigma$, $u_j \in \Sigma^*$ for all $i \in [1, k]$ and $j \in [0, k]$. We call u a proper subword of w if $u \neq w$. We say that a word u is a prefix of a word w if there exists $v \in \Sigma^*$ such that $w = uv$. Given a word $w = a_0 \cdots a_n$, we write $w[i \cdots j]$ to denote the factor $a_i \cdots a_j$. For a $d \in \mathbb{N}$, the sign of d (denoted by $sign(d)$) is defined as $sign(d) = 0$ if $d = 0$ and is 1 otherwise. For all $l \in \mathbb{N}$, we use $\Sigma^{\leq l}$ (resp. Σ^l) to denote the set of words over Σ having length less than or equal to l (resp. exactly equal to l).

2.2 Weighted one-deterministic-counter automata

In this section, we define weighted ODCA, where the weights are from a semiring. However, our results require that the weights come from some field $\mathcal{F}$ (not necessarily finite) except for Section 5, where the weights are from the boolean semiring. First, we define weighted one-counter automata.

▶ **Definition 5.** *A weighted one-counter automaton $\mathcal{A} = (Q, \boldsymbol{\lambda}, \delta_0, \delta_1, \boldsymbol{\eta})$, is defined over an alphabet Σ where, Q is a non-empty finite set of states, $\boldsymbol{\lambda} \in \mathcal{F}^{|Q|}$ is the initial distribution where the ith component of $\boldsymbol{\lambda}$ indicates the initial weight on state $q_i \in Q$, $\delta_0 : Q \times \Sigma \times Q \times \{0, +1\} \to \mathcal{F}$ and $\delta_1 : Q \times \Sigma \times Q \times \{-1, 0, +1\} \to \mathcal{F}$ are the transition functions, and $\boldsymbol{\eta} \in \mathcal{F}^{|Q|}$ is the final distribution, where the ith component of $\boldsymbol{\eta}$ indicates the output weight on state $q_i \in Q$.*

Note that the counter values do not go below zero. Let $p, q \in Q, a \in \Sigma, n \in \mathbb{N}, e \in \{-1, 0, +1\}$, and $s \in \mathcal{F}$. We say $(q, n) \hookrightarrow^{a|s} (p, n + e)$ if $\delta_{sign(n)}(q, a, p, e) = s$. Let $w = a_1 a_2 \cdots a_t \in \Sigma^*$ for some $t \in \mathbb{N}$. For a $q_0 \in Q$ and $n_0 \in \mathbb{N}$, we say $(q_0, n_0) \hookrightarrow^{w|s} (q_t, n_t)$ if for all $i \in [1, t]$, there are $q_i \in Q, n_i \in \mathbb{N}$, $s_i \in \mathcal{F}$ such that $(q_{i-1}, n_{i-1}) \hookrightarrow^{a_i|s_i} (q_i, n_i)$ and $s = \prod_{i=1}^{t} s_i$.

▶ **Definition 6.** *A weighted OCA with counter-determinacy is a weighted one-counter automaton $\mathcal{A} = (Q, \boldsymbol{\lambda}, \delta_0, \delta_1, \boldsymbol{\eta})$ with the following restriction: if $\boldsymbol{\lambda}[i]$ and $\boldsymbol{\lambda}[j]$ are non-zero for some $i, j \in [1, |Q|]$, then for all $w \in \Sigma^*$, if $(q_i, 0) \hookrightarrow^{w|s_1} (p_1, n_1)$ and $(q_j, 0) \hookrightarrow^{w|s_2} (p_2, n_2)$ for some $p_1, p_2 \in Q, n_1, n_2 \in \mathbb{N}$ and $s_1, s_2 \in \mathcal{F}$, then $n_1 = n_2$.*

We present a definition for weighted ODCA, which is an equivalent syntactic model.

▶ **Definition 7.** *A weighted ODCA, $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}))$ is defined over an alphabet Σ where,*

- *C is a non-empty finite set of counter states.*
- *$\delta_0 : C \times \Sigma \to C \times \{0, +1\}$, $\delta_1 : C \times \Sigma \to C \times \{-1, 0, +1\}$ are counter transitions.*
- *$p_0 \in C$ is the start state for the counter structure.*
- *Q is a non-empty finite set of states of the finite state machine.*
- *$\boldsymbol{\lambda} \in \mathcal{F}^{|Q|}$ is the initial distribution where the ith component of $\boldsymbol{\lambda}$ indicates the initial weight on state $q_i \in Q$.*
- *$\Delta : \Sigma \times \{0, 1\} \to \mathcal{F}^{|Q| \times |Q|}$ gives the transition matrix. For all $a \in \Sigma$ and $d \in \{0, 1\}$, the component in the ith row and j^{th} column of $\Delta(a, d)$ denotes the weight on the transition from state $q_i \in Q$ to state $q_j \in Q$ on reading symbol a from counter value n with $sign(n) = d$.*
- *$\boldsymbol{\eta} \in \mathcal{F}^{|Q|}$ is the final distribution, where the ith component of $\boldsymbol{\eta}$ indicates the output weight on state $q_i \in Q$.*

Here, $(C, \delta_0, \delta_1, p_0)$ represents the counter structure and $(Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta})$ represents the finite state machine. Note that δ_0 and δ_1 are deterministic transition functions. The counter structure and the finite state machine run synchronously on any given word. A configuration c of an ODCA is of the form $(\mathbf{x}_\mathsf{c}, p_\mathsf{c}, n_\mathsf{c}) \in \mathcal{F}^{|Q|} \times C \times \mathbb{N}$. We use the notation WEIGHT-VECTOR(c) to denote $\mathbf{x}_\mathsf{c}$, COUNTER-STATE(c) to denote p_c, and COUNTER-VALUE(c) to denote n_c. The initial configuration is $(\boldsymbol{\lambda}, p_0, 0)$. A *transition* is a tuple $\tau = (\iota, d, a, \mathsf{ce}, \mathbb{A}, \theta)$ where $\iota, \theta \in C$ are counter states, $d \in \{0, 1\}$ denotes the sign of the counter value, $a \in \Sigma$, $\mathsf{ce} \in \{-1, 0, 1\}$ is the *counter-effect*, $\mathbb{A} \in \mathcal{F}^{|Q| \times |Q|}$ such that $\Delta(a, d) = \mathbb{A}$, and $\delta_d(\iota, a) = (\theta, \mathsf{ce})$. Given a transition $\tau = (\iota, d, a, \mathsf{ce}, \mathbb{A}, \theta)$ and a configuration $\mathsf{c} = (\mathbf{x}, n, p)$, we denote the application of τ to c as $\tau(\mathsf{c}) = (\mathbf{x}\mathbb{A}, \theta, n + \mathsf{ce})$ if $p = \iota$ and $d = sign(n)$; $\tau(\mathsf{c})$ is undefined otherwise. Note that the counter values are always non-negative.

Consider a sequence of transitions $T = \tau_0 \cdots \tau_\ell$ where $\tau_i = (\iota_i, d_i, a_i, \mathtt{ce}_i, \mathbb{A}_i, \theta_i)$ for all $i \in [0, \ell]$. We denote by $\mathtt{word}(T) = a_0 \cdots a_\ell$ the word labelling it, $\mathtt{we}(T) = \mathbb{A}_0 \cdots \mathbb{A}_\ell$ its weight-effect matrix, and $\mathtt{ce}(T) = \mathtt{ce}_0 + \cdots + \mathtt{ce}_\ell$ its counter-effect. For all $0 \leq i < j \leq \ell$, we use $T_{i \cdots j}$ to denote the sequence of transitions $\tau_i \cdots \tau_j$ and $|T|$ to denote its length $\ell + 1$. We call T *floating* if for all $i \in [0, \ell - 1]$, $d_i = 1$ and *non-floating* otherwise.

A *run* π is an alternating sequence of configurations and transitions denoted as $\pi = \mathtt{c}_0 \tau_0 \mathtt{c}_1 \cdots \tau_{\ell-1} \mathtt{c}_\ell$ such that for every i, $\mathtt{c}_{i+1} = \tau_i(\mathtt{c}_i)$. The word labelling, length, weight-effect, and counter-effect of the run are those of its underlying sequence of transitions. Given a sequence of transitions $T = \tau_0 \cdots \tau_{\ell-1}$ and a configuration $\mathtt{c}$, we denote by $T(\mathtt{c})$ the run (if it is defined) $\mathtt{c}_0 \tau_0 \mathtt{c}_1 \cdots \tau_{\ell-1} \mathtt{c}_\ell$ where $\mathtt{c}_0 = \mathtt{c}$.

For any word w, there is at most one run labelled by w starting from a given configuration $\mathtt{c}_0$. We denote this run $\pi(w, \mathtt{c}_0)$. A run $\pi(w, \mathtt{c}_0) = \mathtt{c}_0 \tau_0 \mathtt{c}_1 \cdots \tau_{\ell-1} \mathtt{c}_\ell$ is also represented as $\mathtt{c}_0 \xrightarrow{w} \mathtt{c}_\ell$. We say $\mathtt{c}_0 \rightarrow^* \mathtt{c}_\ell$ if there is some word w such that $\mathtt{c}_0 \xrightarrow{w} \mathtt{c}_\ell$. For a weighted ODCA $\mathcal{A}$, the accepting weight of w is denoted by $f_{\mathcal{A}}(w, \mathtt{c}) = \boldsymbol{\lambda} \mathtt{we}(\pi(w, \mathtt{c})) \boldsymbol{\eta}^\top$, where $\mathtt{c}$ is the initial configuration of $\mathcal{A}$. Two weighted ODCAs $\mathcal{A}$ and $\mathcal{B}$ are equivalent if for all $w \in \Sigma^*$, $f_{\mathcal{A}}(w, \mathtt{c}) = f_{\mathcal{B}}(w, \mathtt{d})$ where $\mathtt{c}$ and $\mathtt{d}$ are the initial configurations of $\mathcal{A}$ and $\mathcal{B}$ respectively. Let $\mathtt{c}$ and $\mathtt{d}$ be configurations of ODCAs $\mathcal{A}$ and $\mathcal{B}$ respectively. We say that $\mathtt{c} \equiv_l \mathtt{d}$ if and only if for all $w \in \Sigma^{\leq l}$, $f_{\mathcal{A}}(w, \mathtt{c}) = f_{\mathcal{B}}(w, \mathtt{d})$ otherwise $\mathtt{c} \not\equiv_l \mathtt{d}$. An uninitialised weighted ODCA $\mathcal{A}$ is a weighted ODCA without an initial counter state and initial distribution. Weighted automata (WA) is a restricted form of weighted ODCA where the counter value is fixed at zero. The above notions of transitions, runs, acceptance, etc. are used for WA also. Given a weighted ODCA $\mathcal{A}$ and $M \in \mathbb{N}$, we define the M-unfolding weighted automata $\mathcal{A}^M$ as a finite state weighted automaton, where the accepting weight of any word whose run does not encounter counter values greater than M in $\mathcal{A}$ is equal in both $\mathcal{A}$ and $\mathcal{A}^M$. There is a polynomial time procedure to construct $\mathcal{A}^M$.

Consider the weighted ODCA $\mathcal{C}$ recognising the function `prefixAwareDecimal` given in Figure 2a. Here, $\boldsymbol{\lambda} = [1, 0, 0, 0]$ and $\boldsymbol{\eta} = [0, 0, 0, 1]$. The configuration $\mathtt{c}_0 = ([1, 0, 0, 0], p_0, 0)$ is the initial configuration of this machine. Let $w = abaaab$. The run of this machine on the word w can be written as:

$$\pi(w, \mathtt{c}_0) = ([1,0,0,0], p_0, 0) \xrightarrow{a} ([1,0,0,0], p_0, 1) \xrightarrow{b} ([0,1,0,0], p_1, 0) \xrightarrow{a} ([0,0,1,0], p_2, 0)$$
$$\xrightarrow{a} ([0,0,1,1], p_2, 1) \xrightarrow{a} ([0,0,1,2], p_2, 2) \xrightarrow{b} ([0,0,1,6], p_2, 1).$$

The counter-effect of this run is $\mathtt{ce}(\pi(w, \mathtt{c}_0)) = 1$ and the weight-effect matrix is given by

$$\mathtt{we}(\pi(w, \mathtt{c}_0)) = \Delta(a,0)\ \Delta(b,1)\ \Delta(a,0)\ \Delta(a,1)\ \Delta(a,1)\ \Delta(b,1) = \begin{bmatrix} 0 & 0 & 1 & 6 \\ 0 & 0 & 1 & 14 \\ 0 & 0 & 1 & 46 \\ 0 & 0 & 0 & 64 \end{bmatrix}.$$

The accepting weight of the word w is $f_{\mathcal{C}}(w, \mathtt{c}_0) = \boldsymbol{\lambda} \mathtt{we}(\pi(w, \mathtt{c}_0)) \boldsymbol{\eta}^\top = 6$.

3 Reachability problems of weighted ODCA

In this section, we introduce the co-VS reachability and co-VS coverability problems for weighted ODCAs over a field $\mathcal{F}$. We fix a weighted ODCA $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}))$. We use $\mathcal{V} \subseteq \mathcal{F}^{|Q|}$ to denote a vector space and $\overline{\mathcal{V}}$ its complement. Let $S \subseteq C$ be a subset of the set of counter states, $X \subseteq \mathbb{N}$ a set of counter values, and $w \in \Sigma^*$. The notation $\mathtt{c} \xrightarrow{w} \overline{\mathcal{V}} \times S \times X$ denotes the run $\mathtt{c} \xrightarrow{w} \mathtt{d}$ where $\mathtt{d} \in \overline{\mathcal{V}} \times S \times X$ if it exists. We use $\mathtt{c} \xrightarrow{*} \overline{\mathcal{V}} \times S \times X$ to denote that there exists a word $u \in \Sigma^*$ such that $\mathtt{c} \xrightarrow{u} \overline{\mathcal{V}} \times S \times X$.

CO-VS REACHABILITY PROBLEM
INPUT: a weighted ODCA $\mathcal{A}$, an initial configuration c, a vector space $\mathcal{V}$, a set of counter states S, and a counter value m.
OUTPUT: *Yes*, if there exists a run $\mathsf{c} \xrightarrow{*} \overline{\mathcal{V}} \times S \times \{m\}$ in $\mathcal{A}$. *No*, otherwise.

CO-VS COVERABILITY PROBLEM
INPUT: a weighted ODCA $\mathcal{A}$, an initial configuration c, a vector space $\mathcal{V}$, and a set of counter states S.
OUTPUT: *Yes*, if there exists a run $\mathsf{c} \xrightarrow{*} \overline{\mathcal{V}} \times S \times \mathbb{N}$ in $\mathcal{A}$. *No*, otherwise.

Unlike the co-VS reachability problem, the final configuration's counter value is not considered part of the input for co-VS coverability problem. We assume that the vector space $\mathcal{V} \subseteq \mathcal{F}^{|Q|}$ is provided by giving a basis. We call $z \in \Sigma^*$ a *witness* of $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$ if $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times X$. Furthermore, z is called a *minimal* witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$ if for all $u \in \Sigma^*$ with $\mathsf{c} \xrightarrow{u} \overline{\mathcal{V}} \times S \times X$, $|u| \geq |z|$.

In the upcoming subsection, we give some interesting properties of minimal witnesses. In Section 3.2, we provide a pseudo-pumping lemma which helps us show that co-VS reachability and co-VS coverability are in P if the counter values are given in unary notation. Finally, in Section 3.3, we demonstrate that the lexicographically minimal witness has a canonical form. In the following subsections, $\mathcal{V}$ denotes a vector space, c a configuration, S a subset of counter states, and $X \subseteq \mathbb{N}$. We also denote by $K = |Q| \cdot |C|$, where C is the set of counter states, and Q is the set of states of the finite state machine.

3.1 Minimal witness and its properties

The following observation helps in breaking down the reachability problem into sub-problems. If $z \in \Sigma^*$ is a minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$, then for every z_1, z_2 such that $z = z_1 z_2$, there is a vector space $\mathcal{U}$ such that z_1 is a minimal witness for $(\mathsf{c}, \overline{\mathcal{U}}, \{p\}, \{n\})$ where p is the counter state and n is the counter value reached after reading z_1 from c.

▶ **Observation 8.** *Consider arbitrary $z, z_1, z_2 \in \Sigma^*$ such that $z = z_1 z_2$. Let $\mathsf{d} = (\mathbf{x}_d, p_d, n_d)$ and $\mathsf{e} = (\mathbf{x}_e, p_e, n_e)$ be configurations such that $\mathsf{c} \xrightarrow{z_1} \mathsf{d} \xrightarrow{z_2} \mathsf{e}$ and $\mathbb{A} \in \mathcal{F}^{|Q| \times |Q|}$ be such that $\mathbf{x}_d \mathbb{A} = \mathbf{x}_e$. If z is a minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$, then z_1 is a minimal witness for $(\mathsf{c}, \overline{\mathcal{U}}, \{p_d\}, \{n_d\})$, where $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{|Q|} \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}$.*

We aim to show that the length of a minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$ is polynomially bounded. The following lemma shows that if the counter values are polynomially bounded during the run of a minimal witness, then its length is also polynomially bounded.

▶ **Lemma 9.** *Let $z \in \Sigma^*$ be a minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$. If the number of distinct counter values encountered during the run $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times X$ is t, then $|z| \leq t \cdot K$.*

It now suffices to show that the counter values encountered during the run of a minimal witness are polynomially bounded.

3.2 Pseudo-pumping lemma

The pseudo-pumping lemma is a valuable tool in our analysis, allowing us to pump up or down a sufficiently long word while maintaining the reachability conditions.

▶ **Lemma 10** (pseudo-pumping lemma). *Let $m, R \in \mathbb{N}$, be such that* COUNTER-VALUE$(\mathtt{c}) = m$ *and $z \in \Sigma^*$ be such that $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run, and the maximum counter value encountered during this run is $m + R$. If $R > K^2$, then there exists $z_{sub}, z_{sup} \in \Sigma^*$ such that the following hold:*

1. *there exist $x, y, u, v, w \in \Sigma^*$ such that $z = xyuvw, z_{sub} = xuw$, $\mathtt{c} \xrightarrow{z_{sub}} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run, and the counter values encountered during this run are less than $m + R$, and*
2. *there exist $x, y, u, v, w \in \Sigma^*$ such that $z = xyuvw, z_{sup} = xy^2uv^2w$, $\mathtt{c} \xrightarrow{z_{sup}} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run, and the maximum counter value encountered in this run exceeds $m + R$.*

Proof. Let $z \in \Sigma^*$ be a witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ and $\mathtt{e} \in \overline{\mathcal{V}} \times S \times \{m\}$ be such that $\mathtt{c} \xrightarrow{z} \mathtt{e}$ is a floating run, and the maximum counter value encountered in this run be $m + R$ where $R > K^2$. Let COUNTER-VALUE$(\mathtt{c}) = m$. There exist $z_1, z_2 \in \Sigma^*$ and configuration $\mathtt{f}$ such that $z = z_1 z_2$ and $\mathtt{c} \xrightarrow{z_1} \mathtt{f} \xrightarrow{z_2} \mathtt{e}$, where COUNTER-VALUE$(\mathtt{f}) = m + R$ (see Figure 3).

Let $\mathtt{c}_1 = \mathtt{c}$ and $\pi = \mathtt{c}_1 \tau_1 \mathtt{c}_2 \cdots \tau_{\ell-1} \mathtt{c}_\ell$ denote the run on word z from the configuration $\mathtt{c}_1$ and $T = \tau_1 \tau_2 \cdots \tau_{\ell-1}$ the sequence of transitions of π. For any $i \in [0, R]$, we denote by l_i and d_i the indices such that a configuration with counter value $m + i$ is encountered for the last (resp. first) time before (resp. after) reaching counter value $m + R$ in π. That is, COUNTER-VALUE$(\mathtt{c}_{l_i})$ = COUNTER-VALUE$(\mathtt{c}_{d_i}) = m + i$, and for any j where $l_i < j < d_i$, COUNTER-VALUE$(\mathtt{c}_j) > m + i$. To simplify the notation, we denote by $\mathtt{g}_i = \mathtt{c}_{l_i}$ and $\mathtt{g}'_i = \mathtt{c}_{d_i}$.

Consider the pairs of configurations $(\mathtt{g}_1, \mathtt{g}'_1), (\mathtt{g}_2, \mathtt{g}'_2), \ldots, (\mathtt{g}_R, \mathtt{g}'_R)$. Since $R > (|Q| \cdot |C|)^2$, by the Pigeonhole principle, there exist two counter states p, q, and a set of indices $I \subseteq [0, R]$ where $|I| = |Q|^2 + 1$ such that for all $h \in I$, COUNTER-STATE$(\mathtt{g}_h) = p$ and COUNTER-STATE$(\mathtt{g}'_h) = q$. For all $j \in I$, let $u_j, v_j, w_j \in \Sigma^*$ be such that $\mathtt{c}_1 \xrightarrow{u_j} \mathtt{g}_j \xrightarrow{v_j} \mathtt{g}'_j \xrightarrow{w_j} \mathtt{e}$. We use the following shorthand for any configuration $\mathtt{g}$: $\mathbf{x}_{\mathtt{g}}$ = WEIGHT-VECTOR$(\mathtt{g})$. For all $j \in I$, let matrix $\mathbb{A}_j$ and $\mathbb{B}_j$ be such that $\mathbf{x}_{\mathtt{g}'_j} = \mathbf{x}_{\mathtt{g}_j} \mathbb{A}_j$ and $\mathbf{x}_{\mathtt{e}} = \mathbf{x}_{\mathtt{g}'_j} \mathbb{B}_j$. Since $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{V}}$, for all $j \in I$, $\mathbf{x}_{\mathtt{g}_j} \mathbb{A}_j \mathbb{B}_j \in \overline{\mathcal{V}}$. Let $r = |Q|^2 + 1$, and $i_1 < i_2 < \cdots < i_r$ be the indices in I. We prove the two cases separately.

1. Consider the sequence of matrices $\mathbb{A}_{i_r}, \mathbb{A}_{i_{r-1}}, \ldots, \mathbb{A}_{i_1}$. Since there can be at most $|Q|^2$ independent matrices, there exists $k \in [1, r]$ such that $\mathbb{A}_{i_k}$ is a linear combination of $\mathbb{A}_{i_r}, \ldots, \mathbb{A}_{i_{k+1}}$. Hence, there exists $h \in \{i_r, \ldots, i_{k+1}\}$ such that $\mathbf{x}_{\mathtt{g}_{i_k}} \mathbb{A}_h \mathbb{B}_{i_k} \in \overline{\mathcal{V}}$. Let $z_{sub} = u_{i_k} v_h w_{i_k}$. It is easy to observe that z_{sub} is a subword of z as mentioned in the lemma. To conclude the proof, it now suffices to show that z_{sub} is a witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ and the counter values encountered during the run $\mathtt{c} \xrightarrow{z_{sub}} \mathtt{h}$ are less than $m + R$. Consider the floating run $\mathtt{g}_h \xrightarrow{v_h} \mathtt{g}'_h$. From the choice of $\mathtt{g}_h$ and $\mathtt{g}'_h$ we know that COUNTER-VALUE$(\mathtt{g}_h)$ = COUNTER-VALUE$(\mathtt{g}'_h) = m + h$ and for all j where $l_h < j < d_h$, COUNTER-VALUE$(\mathtt{c}_j) > m + h$. Since COUNTER-STATE$(\mathtt{g}_h)$ = COUNTER-STATE$(\mathtt{g}_{i_k})$, $\pi(v_h, \mathtt{g}_{i_k})$ is also a floating run $\mathtt{g}_{i_k} \xrightarrow{v_h} \mathtt{d}$ such that COUNTER-STATE$(\mathtt{g}'_h)$ = COUNTER-STATE$(\mathtt{d})$, COUNTER-VALUE$(\mathtt{g}_{i_k})$ = COUNTER-VALUE$(\mathtt{d}) = m + i_k < m + h$, and the minimum and maximum counter values encountered in the run is $m + i_k$ and $m + R - (h - i_k)$ respectively (see Figure 3). Furthermore, $\mathbf{x}_d = \mathbf{x}_{\mathtt{g}_{i_k}} \mathbb{A}_h$. Since COUNTER-STATE$(\mathtt{g}'_{i_k})$ = COUNTER-STATE$(\mathtt{g}'_h)$, we get that COUNTER-STATE$(\mathtt{g}'_{i_k})$ = COUNTER-STATE$(\mathtt{d})$. Moreover, since COUNTER-VALUE$(\mathtt{g}'_{i_k})$ = COUNTER-VALUE$(\mathtt{g}_{i_k})$, we have COUNTER-VALUE$(\mathtt{g}'_{i_k})$ = COUNTER-VALUE$(\mathtt{d})$. Therefore, $\pi(w_{i_k}, \mathtt{d})$ is the run $\mathtt{d} \xrightarrow{w_{i_k}} \mathtt{h}$ where $\mathbf{x}_{\mathtt{h}} = \mathbf{x}_{\mathtt{d}} \mathbb{B}_{i_k}$ and hence $\mathbf{x}_{\mathtt{h}} = \mathbf{x}_{\mathtt{g}_{i_k}} \mathbb{A}_h \mathbb{B}_{i_k} \in \overline{\mathcal{V}}$. This concludes that z_{sub} is a witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ and satisfies the properties mentioned in the lemma.

2. Consider the sequence of matrices: $\mathbb{A}_{i_1}, \mathbb{A}_{i_2}, \ldots, \mathbb{A}_{i_r}$. Note that the matrices are ordered in reverse compared to the ordering in the previous case. By following a similar argument, we get a word z_{sup} satisfying the required properties. ◀

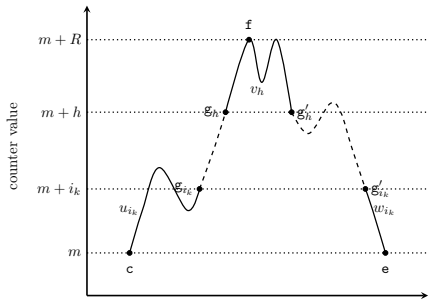


Figure 3 The figure shows the floating run from a configuration c with COUNTER-VALUE(c) $= m$ to a configuration $\mathsf{e} = (\mathbf{x}, p, m)$ such that $\mathbf{x} \in \overline{\mathcal{U}}$. Configurations g_{i_k} and g_h (resp. g'_{i_k} and g'_h) are where the counter values $m + i_k$ and $m + h$ are encountered for the last (resp. first) time before (resp. after) reaching $m + R$. Also, COUNTER-STATE(g_{i_k}) $=$ COUNTER-STATE(g_h) and COUNTER-STATE(g'_h) $=$ COUNTER-STATE(g'_{i_k}). The dashed line denotes the part of the run that can be removed to get a shorter witness for $(\mathsf{c}, \overline{\mathcal{U}}, \{p\}, \{n\})$.

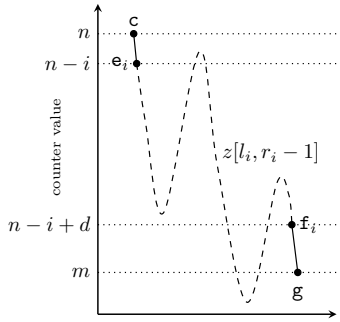


Figure 4 The figure shows the floating run from a configuration c with COUNTER-VALUE(c) $= n$ to a configuration $\mathsf{g} = (\mathbf{x}, p, m)$ such that $\mathbf{x} \in \overline{\mathcal{V}}$. The points e_i and f_i denotes the configurations where the counter values $n - i$ and $n - i + d$ are encountered for the first (resp. last) time during this run. The dashed line represents the part of the run due to factor $z[l_i, r_i - 1]$ and has a counter effect d.

It is important to note that we do not end up in the same configuration while pumping up/down, but we ensure that we reach a configuration with the same counter state, counter value, and whose weight vector is in the complement of the given vector space.

Now, we prove that for any run (it need not necessarily be a floating run) of a minimal reachability witness z for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, the maximum counter value encountered during the run $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is bounded by a polynomial in the number of states of the machine, and the initial and final counter values. This can be achieved by iteratively applying Lemma 10 on the run of the minimal witness (refer Figure 6) and using Observation 8 and Lemma 9.

▶ **Corollary 11.** *If $z \in \Sigma^*$ is a minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, then*

1. *the maximum counter value encountered during the run $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is less than $max(\text{COUNTER-VALUE}(\mathsf{c}), m) + K^2$, and*
2. *$|z| \leq K^3 + max(\text{COUNTER-VALUE}(\mathsf{c}), m) \cdot K$.*

The following lemma (depicted in Figure 5) helps us show that the length of a minimal witness for co-VS coverability is polynomially bounded in the number of states.

▶ **Lemma 12** (cut lemma)**.** *Let $z \in \Sigma^*$ be a witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, where c is a configuration with COUNTER-VALUE(c) $= n$ for some $n \in \mathbb{N}$, and $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run for some $m \in \mathbb{N}$. If $m - n > K$, then there exists $z_{sub} \in \Sigma^*$ such that z_{sub} is a subword of z, $\mathsf{c} \xrightarrow{z_{sub}} \overline{\mathcal{V}} \times S \times \{m'\}$ is a floating run and $m' - n < m - n$.*

Now, we prove that the co-VS reachability and co-VS coverability problems of weighted ODCA are in P by demonstrating a small model property. We have already established using Lemma 10, Corollary 11, and Lemma 12 that the maximum and minimum counter values encountered during the run of the minimal witness do not exceed some polynomial bound. This, in turn, implies a polynomial bound on the length of the witness by Lemma 9. As a result, we get the following theorem.

▶ **Theorem 13.** *The co-VS reachability and co-VS coverability problems for weighted ODCA can be decided in polynomial time when the counter values are given in unary notation.*

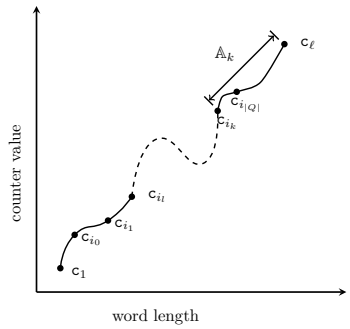


Figure 5 The figure shows a run from configuration c_1 to $\mathsf{c}_\ell = (\mathbf{x}_{\mathsf{c}_\ell}, p_{\mathsf{c}_\ell}, n_{\mathsf{c}_\ell})$ such that $\mathbf{x}_{\mathsf{c}_\ell} \in \overline{\mathcal{V}}$. The configurations c_{i_l} and c_{i_k} are where the counter values $n_{\mathsf{c}_{i_l}}$ and $n_{\mathsf{c}_{i_k}}$ are encountered for the last time. Also the configurations c_{i_l} and c_{i_k} have the same counter state. The dashed line is the part that can be removed to get a shorter witness for $(\mathsf{c}, \overline{\mathcal{V}}, \{p_{\mathsf{c}_\ell}\}, \mathbb{N})$.

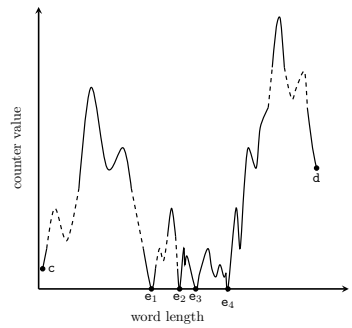


Figure 6 The figure shows a run from configuration c to $\mathsf{d} = (\mathbf{x}_\mathsf{d}, p_\mathsf{d}, n_\mathsf{d})$ such that $\mathbf{x}_\mathsf{d} \in \overline{\mathcal{V}}$. Configurations $\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3, \mathsf{e}_4$ are where the counter value zero is encountered during the run. The dashed lines denote the parts that can be removed to obtain a shorter witness for $(\mathsf{c}, \overline{\mathcal{V}}, \{p_\mathsf{d}\}, \{n_\mathsf{d}\})$.

3.3 Lexicographically minimal witness

This section will show that the lexicographically minimal witness has a distinct structure.

We assume a total order on the symbols in Σ. Given two words $u, v \in \Sigma^*$, we say that u precedes v in the *(length) lexicographical ordering* if $|u| < |v|$ or if $|u| = |v|$ and there exists an $i \in [0, |u| - 1]$ such that $u[0, i-1] = v[0, i-1]$ and $u[i]$ precedes $v[i]$ in the total ordering assumed on Σ. A word $z \in \Sigma^*$ is called the lexicographically minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, if $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ and for all $u \in \Sigma^* \setminus \{z\}$ with $\mathsf{c} \xrightarrow{u} \overline{\mathcal{V}} \times S \times \{m\}$, z precedes u in the lexicographical ordering. We show that the lexicographically minimal witness z for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$ has a canonical form. First, we prove this for floating runs.

▶ **Lemma 14.** *There exist polynomials $p_1 : \mathbb{N} \to \mathbb{N}$, and $p_2 : \mathbb{N}^2 \to \mathbb{N}$ such that, if $z \in \Sigma^*$ is the lexicographically minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$ and $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run, then there exist $u, y, w \in \Sigma^*$ and $r \in \mathbb{N}$ such that $z = uy^r w$ and the following are true:*

1. *$|uyw| \leq p_1(K)$, and*
2. *$r \leq p_2(K, |\textsc{counter-value}(\mathsf{c}) - m|)$.*

Proof. Let z be the lexicographically minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, and $\mathsf{g} \in \overline{\mathcal{V}} \times S \times \{m\}$ such that $\mathsf{c} \xrightarrow{z} \mathsf{g}$ is a floating run. Let n be such that $\textsc{counter-value}(c) = n$. We consider the case $n > m$. The case where $m \geq n$ is analogous. Let $t = n - m$.

▷ Claim 1. $|z| \leq 2K^3 + t \cdot K$.

Proof. From Point 1 of Lemma 10, it follows that the maximum counter value during the run $\mathsf{c} \xrightarrow{z} \mathsf{g}$ is less than $n + K^2$. By a symmetric argument, it follows that the minimum counter value during the run is greater than $m - K^2$. Hence, there are at most $t + 2K^2$ distinct counter values during the run. From Lemma 9 it follows that $|z| \leq 2K^3 + t \cdot K$. ◁

If $t \leq K^2$, then from Claim 1, we get that $|z| \leq 3K^3$, and the lemma is trivially true. Let us assume $t > K^2$ and let $d = K^2 - t$. Note that d is a negative number. Let $\mathsf{c}_1 = \mathsf{c}$ and $\pi(z, \mathsf{c}_1) = \mathsf{c}_1 \tau_1 \mathsf{c}_2 \cdots \tau_{\ell-1} \mathsf{c}_\ell$ denote the run on word z from c. For any $i \in [0, K^2]$, we denote by l_i the index such that the counter value $n - i$ is encountered for the first time, and r_i the index such that the counter value $n - i + d$ is encountered for the last time in

$\pi(z, \mathsf{c}_1)$ (see Figure 4). Let $I = \{(l_i, r_i)\}_{i \in [0, K^2]}$ be the set of these pairs of indices, and let $W = \{z[l, r-1] \mid (l, r) \in I\}$ be the set of corresponding factors. Note that $|I| > K^2$. We argue that these factors $z[l_i, r_i - 1]$ for $i \in [0, K^2]$ need not all be distinct.

▷ Claim 2. $|W| \leq K^2$.

Proof. Assume for contradiction that $|W| > (|Q| \cdot |C|)^2$. Since the number of counter states is $|C|$, by Pigeonhole principle there exists $Y \subseteq I$ with $|Y| = |Q|^2 + 1$ such that for all $(l, r), (l', r') \in Y$, configurations c_l and $\mathsf{c}_{l'}$ have the same counter state, configurations c_r and $\mathsf{c}_{r'}$ have the same counter state, and $z[l, r-1] \neq z[l', r'-1]$. We say $(l, r) < (l', r')$ if $z[l, r-1]$ precedes $z[l', r'-1]$ in the lexicographical order. Therefore, the elements in Y have an ordering as follows: $(l_0, r_0) < (l_1, r_1) < \cdots < (l_{|Q|^2}, r_{|Q|^2})$. For any configuration $\mathtt{h}$, let $\mathbf{x}_{\mathtt{h}} = \textsc{weight-vector}(\mathtt{h})$. For all $i \in [0, |Q|^2]$, let $u_i = z[1, l_i - 1], x_i = z[l_i, r_i - 1], w_i = z[r_i, \ell - 1]$, configurations $\mathtt{e}_i, \mathtt{f}_i$ be such that $\mathtt{c} \xrightarrow{u_i} \mathtt{e}_i \xrightarrow{x_i} \mathtt{f}_i \xrightarrow{w_i} \mathtt{g}$ and matrices $\mathbb{A}_i, \mathbb{M}_i, \mathbb{B}_i$ be such that $\mathbf{x}_{\mathtt{e}_i} = \mathbf{x}_{\mathtt{c}} \mathbb{A}_i$, $\mathbf{x}_{\mathtt{f}_i} = \mathbf{x}_{\mathtt{e}_i} \mathbb{M}_i$, $\mathbf{x}_{\mathtt{g}} = \mathbf{x}_{\mathtt{f}_i} \mathbb{B}_i$.

We know that for all $k \in [0, |Q|^2]$, $\mathbf{x}_{\mathtt{c}} \mathbb{A}_k \mathbb{M}_k \mathbb{B}_k \in \overline{\mathcal{V}}$. Consider the sequence of matrices $\mathbb{M}_0, \mathbb{M}_1, \cdots, \mathbb{M}_{|Q|^2}$. Since there can be at most $|Q|^2$ independent matrices, we get that there exists $i \in [0, |Q|^2]$ such that $\mathbb{M}_i$ is a linear combination of $\mathbb{M}_0, \ldots, \mathbb{M}_{i-1}$. Hence, we get that there exists a j where $j < i$ such that $\mathbf{x}_{\mathtt{c}} \mathbb{A}_i \mathbb{M}_j \mathbb{B}_i \in \overline{\mathcal{V}}$. Since $x_j = z[l_j, r_j - 1]$ precedes $x_i = z[l_i, r_i - 1]$, the word $u_i x_j w_i$ precedes z in the lexicographical ordering. Therefore the run $\pi(u_i x_j w_i, \mathtt{c})$ contradicts the lexicographical minimality of z. ◁

Since $|W| \leq K^2$ and $|I| > K^2$, there exists $i, j \in [0, K^2]$, with $i < j$ and $x \in \Sigma^*$ such that $(l_i, r_i) \in I, (l_j, r_j) \in I$ and $x = z[l_i, r_i - 1] = z[l_j, r_j - 1]$ (see Figure 7). Let $u_1, w_1, u_2, w_2 \in \Sigma^*$ such that $z = u_1 x w_1 = u_2 x w_2$. Since $u_1 \neq u_2$, either u_1 is a prefix of u_2 or u_2 a prefix of u_1. Without loss of generality, let us assume u_1 is a prefix of u_2. Therefore, there exists $v \in \Sigma^*$ such that $u_2 = u_1 v$. Let $\mathtt{e}$ be a configuration such that $\mathtt{c} \xrightarrow{u_1} \mathtt{e}$.

▷ Claim 3. $|u_1|, |v|, |w_1| \leq 3K^3$.

Proof. Consider the set I. For any $i, j \in [0, K^2]$, the difference between the counter values of configurations c_{l_i} and c_{l_j} and the difference between the counter values of the configurations c_{r_j} and c_{r_i} is at most $K^2 + 1$. Therefore the counter-effect of u_2, w_2, and v can be at most K^2. Since $\pi(v, \mathtt{e})$ is a floating run from Claim 1, we get that $|v| \leq 3K^3$. By similar arguments, the counter-effect of u_1 and w_1 can be at most K^2, and again by Claim 1, we get that their lengths are at most $3K^3$. ◁

▷ Claim 4. There exist $v' \in \Sigma^*$ and $r \in [0, K^3 + t \cdot K]$ such that $x = v^r v'$ with $|v'| \leq |v|$.

Proof. Let $r \in \mathbb{N}$ be the largest number such that x is of the form $v^r v'$ for some $v' \in \Sigma^*$ (see Figure 7). We know that $z = u_2 x w_2$ and $u_2 = u_1 v$. Therefore, $z = u_1 v x w_2 = u_1 v v^r v' w_2 = u_1 v^r v v' w_2$. Furthermore, $z = u_1 x w_1 = u_1 v^r v' w_1$. Now since $u_1 v^r v v' w_2 = u_1 v^r v' w_1$, we get that $v v' w_2 = v' w_1$. Hence, if $|v'| \geq |v|$, then v is a prefix of v'. This is a contradiction since r was chosen to be the largest number such that x is of the form $v^r v'$.

To show the bound on the value r, we observe the following. We know that the counter effect of the run $\pi(x, \mathtt{e})$ is d. Therefore from Claim 1, we get that $|x| \leq 2K^3 + |d| \cdot K$. Hence, $r \leq 2K^3 + |d| \cdot K$. ◁

From Claim 4 and Claim 3, we get that $|u_1 v v' w_1| \leq 12K^3$ and $z = u_1 v^r v' w_1$ for some $r \in [0, 2K^3 + |d| \cdot K)]$. ◀

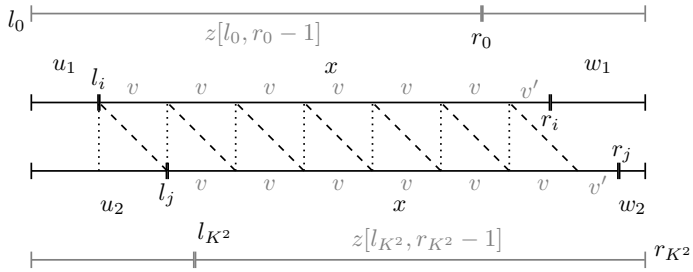


Figure 7 The figure shows the factorisation of a word $z = u_1xw_1 = u_2xw_2$, where $x = z[l_i, r_i - 1] = z[l_j, r_j - 1]$, and $u_1 \neq u_2$. The factor v is a prefix of x such that $u_2 = u_1v$. The word z can be written as $u_1v^iv'w_2$ for some $i \in \mathbb{N}$ and v' prefix of v. For $k \in [0, K^2]$, l_k is the index such that the counter value $n - k$ is encountered for the first time and r_k the index such that the counter value $n - k + d$ is encountered for the last time during the run $\mathtt{c} \xrightarrow{z} \mathtt{g}$.

We now establish that the lexicographically minimal witness z (whose run need not be floating) for a co-VS reachability problem has the form $uy_1^{r_1}vy_2^{r_2}w$. Here, lengths of the words u, y_1, y_2, v, and w are polynomially bounded in the number of states, and r_1 and r_2 are polynomial values dependent on the number of states and the input counter values.

▶ **Lemma 15** (special-word lemma). *If $z \in \Sigma^*$ is the lexicographically minimal witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$, then there exists $u, y_1, v_1, v_2, v_3, y_2, w \in \Sigma^*$ and $r_1, r_2 \in \mathbb{N}$ such that $z = uy_1^{r_1}vy_2^{r_2}w$ and the following are true:*

1. *$|uy_1vy_2w|$ is polynomially bounded in the number of states of the machine.*
2. *r_1 and r_2 are polynomially bounded in the number of states of the machine, m, and* COUNTER-VALUE$(\mathtt{c})$.

4 Equivalence of weighted ODCA

In this section, we present a polynomial time algorithm to decide the equivalence of two weighted ODCAs whose weights come from a fixed field. The techniques developed in the previous section in conjunction with those presented in Valiant and Paterson [24], and Böhm et al. [3] for deterministic real-time OCA give us the algorithm. Here, we give a proof sketch of Theorem 3.

In the remainder of this section, we fix two non-equivalent weighted ODCAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over an alphabet Σ and a field $\mathcal{F}$. We fix a minimal word z (also called *witness*) that distinguishes them. We denote a configuration pair as $\mathtt{h}_i = \langle \mathtt{c}_i, \mathtt{d}_i \rangle$ where $\mathtt{c}_i$ is a configuration of $\mathcal{A}_1$ and $\mathtt{d}_i$ is a configuration of $\mathcal{A}_2$. We denote by $\Pi = \mathtt{h}_0\tau_0\mathtt{h}_1 \cdots \tau_{\ell-1}\mathtt{h}_{|z|}$ the synchronisation of runs over z in $\mathcal{A}_1$ and $\mathcal{A}_2$ from their initial configurations, where $\mathtt{h}_i$ are pairs of configurations and τ_i are pairs of transitions. The main idea to prove Theorem 3 is to show that the length of z is polynomially bounded in the size of the two weighted ODCAs.

▶ **Lemma 16.** *There is a polynomial* $\mathrm{poly}_0 : \mathbb{N} \to \mathbb{N}$ *such that if two weighted* ODCAs *$\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent, then there exists a witness z such that the counter values encountered during Π are less than* $\mathrm{poly}_0(max(|\mathcal{A}_1|, |\mathcal{A}_2|))$.

Lemma 16 allows to show that the length of the witness z is bounded by a polynomial $\mathrm{poly}_1(max(|\mathcal{A}_1|, |\mathcal{A}_2|))$. Thus we can reduce the equivalence problem of weighted ODCA over fields to that of weighted automata over fields (which is in P [23]) by "simulating" the runs of weighted ODCAs $\mathcal{A}_1$ and $\mathcal{A}_2$ up to length $\mathrm{poly}_1(max(|\mathcal{A}_1|, |\mathcal{A}_2|))$ using two weighted automata that are unfolding of the weighted ODCAs upto counter value $\mathrm{poly}_1(max(|\mathcal{A}_1|, |\mathcal{A}_2|))$.

The rest of this section is dedicated to proving Lemma 16. Following Böhm et al. [3], we define a partition of the set of configuration pairs to facilitate this. We partition the set of configuration pairs into three: initial space, belt space, and background space (see Figure 8). Given a configuration $\mathtt{c}$, we use $n_{\mathtt{c}}$ to denote COUNTER-VALUE($\mathtt{c}$).

- *initial space*: All configuration pairs $\langle \mathtt{c}, \mathtt{d} \rangle$ such that $n_{\mathtt{c}}, n_{\mathtt{d}} < \text{poly}_2(max(|\mathcal{A}_1|, |\mathcal{A}_2|))$.
- *belt space*: Let α, β be co-prime numbers whose values are bounded by a polynomial in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$. A belt of slope $\frac{\alpha}{\beta}$ consists of those configuration pairs $\langle \mathtt{c}, \mathtt{d} \rangle$ outside the initial space that satisfies $|\alpha.n_{\mathtt{c}} - \beta.n_{\mathtt{d}}| \leq \text{poly}_3(max(|\mathcal{A}_1|, |\mathcal{A}_2|))$. The belt space contains all configuration pairs $\langle \mathtt{c}, \mathtt{d} \rangle$ that are inside belts with slope $\frac{\alpha}{\beta}$.
- *background space*: All remaining configuration pairs.

These partitions are indexed on two carefully chosen polynomials poly_2 and poly_3, so that all belts are disjoint outside the initial space.

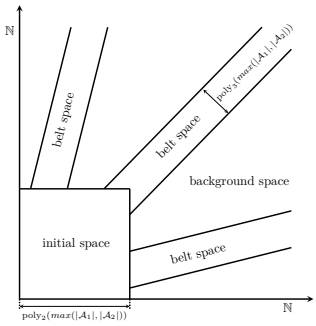


Figure 8 Projection of configuration space.

To prove Lemma 16, there are two cases to consider: either there is no background space point in Π, or there is a background space point in Π.

Case 1: When there is no background space point in Π

Since there is no background space point in Π, all the points in Π are either in the initial or belt space. By definition, the counter values of configuration pairs inside the initial space are bounded by $\text{poly}_2(max(|\mathcal{A}_1|, |\mathcal{A}_2|))$. Now, we look at the sub-run of Π inside the belt space. If a sub-run of Π enters and exits a belt from the initial space or if Π ends inside a belt, then we show that the counter values encountered during that belt visit are polynomially bounded. For this proof, it is crucial that the belts are disjoint.

Consider a pair of sub-run inside a belt. The counter values of $\mathcal{A}_1$ and $\mathcal{A}_2$ are related by a linear expression. In particular, given the counter value of $\mathcal{A}_1$, the counter value of $\mathcal{A}_2$ can only have polynomial many choices. Hence an ODCA $\mathcal{A}_3$ can simulate this run of the pair inside a belt by tracking the counter value of $\mathcal{A}_1$ in its counter and the other counter value using finitely many states. We use co-Vs reachability/coverability to show that if a sub-run of this ODCA $\mathcal{A}_3$ reaches counter values higher than some polynomial in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$, then there exists a shorter witness (contradicting minimality). We achieve this by applying pseudo-pumping lemma (Lemma 10) and cut lemma (Lemma 12) on this sub-run. Hence, we get that the pair of runs of the minimal witness cannot reach counter values higher than some polynomial bound if it does not enter the background space.

Case 2: When there is a background space point in Π

We now consider the case where the witness ultimately enters the background space. Using co-VS reachability, we prove that the counter values encountered during Π till the first background space point are polynomially bounded. We also show that the length of the remaining run is polynomially bounded in the number of states of the machines.

To that end, we need the notion of *underlying uninitialised weighted automaton*. Roughly speaking, an underlying uninitialised weighted automaton of an ODCA $\mathcal{A}$ is the uninitialised weighted automaton $\mathrm{U}(\mathcal{A})$ that is syntactically equivalent to $\mathcal{A}$ without zero tests. In other words, the transition function of $\mathrm{U}(\mathcal{A})$ will be determined by the transition functions of $\mathcal{A}$ for positive counter values. Floating runs of $\mathcal{A}$ are isomorphic to runs of this weighted automaton $\mathrm{U}(\mathcal{A})$. Given $k \in \mathbb{N}$, a configuration $\mathtt{c}$ of a weighted ODCA $\mathcal{A}$ is said to be k-equivalent to a configuration $\bar{\mathtt{c}}$ of an uninitialised weighted automaton $\mathcal{B}$, if for all $w \in \Sigma^{\leq k}$, $f_{\mathcal{A}}(w, \mathtt{c}) = f_{\mathcal{B}}(w, \bar{\mathtt{c}})$. We say that $\mathtt{c}$ is not k-equivalent to $\bar{\mathtt{c}}$ otherwise.

We consider the uninitialised weighted automaton $\mathcal{B}$, which is a disjoint union of $\mathrm{U}(\mathcal{A}_1)$ and $\mathrm{U}(\mathcal{A}_2)$. This gives us a single automaton with which we can compare the configurations of $\mathcal{A}_1$ and $\mathcal{A}_2$. For $i \in \{1, 2\}$, let C_i be the set of counter states of $\mathcal{A}_i$. For all $p \in C_i$ and $m < |\mathcal{B}|$, we define the sets $\mathcal{W}_i^{p,m}$. The set $\mathcal{W}_i^{p,m}$ contains vectors $\mathbf{x}$ such that the configuration $(\mathbf{x}, p, m)$ of $\mathcal{A}_i$ is $|\mathcal{B}|$-equivalent to some configuration of $\mathcal{B}$. The set $\overline{\mathcal{W}}_i^{p,m}$ is the complement of $\mathcal{W}_i^{p,m}$. For any $i \in \{1, 2\}$, $p \in C_i$ and $m < |\mathcal{B}|$, the set $\mathcal{W}_i^{p,m}$ is a vector space. The distance of a configuration $\mathtt{c}$ of $\mathcal{A}_i$ (denoted as $\mathrm{dist}_{\mathcal{A}_i}(\mathtt{c})$) is the length of a minimal word that takes you from $\mathtt{c}$ to a configuration $(\mathbf{x}, p, m)$ for some $m < |\mathcal{B}|$ and $p \in C_i$ such that $\mathbf{x} \in \overline{\mathcal{W}}_i^{p,m}$. Given two configurations $\mathtt{c}$, $\mathtt{d}$ of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively, if $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d})$, then $\mathtt{c} \not\equiv \mathtt{d}$.

By special word lemma (Lemma 14), the lexicographically minimal reachability witness has a special form. This is used to show that if a configuration $\mathtt{c}$ of an ODCA $\mathcal{A}$ has finite distance, then $\mathrm{dist}_{\mathcal{A}}(\mathtt{c}) = \frac{a}{b}\textsc{counter-value}(\mathtt{c}) + t$, where $a, b, t \in \mathbb{N}$ and are polynomially bounded in $|\mathcal{A}|$. This helps us in proving that configuration pairs outside the initial space having equal distance lie inside a belt. Therefore, the background space points either have unequal or infinite distances. Similar to that in [3], we can show that the length of the run Π in the background space is polynomially bounded in $|\mathcal{A}_1|, |\mathcal{A}_2|$, and the counter values of the first background point in Π. The following lemma bounds the counter values of the first configuration pair in the background space, if it exists, during the run Π.

▶ **Lemma 17.** *If $\mathtt{h}_j = \langle \mathtt{c}_j, \mathtt{d}_j \rangle$ is the first background point in Π then, $\textsc{counter-value}(\mathtt{c}_j)$ and $\textsc{counter-value}(\mathtt{d}_j)$ are polynomially bounded in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$.*

Proof sketch. Since $\mathtt{h}_j = \langle \mathtt{c}_j, \mathtt{d}_j \rangle$ is a background point in Π, either $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j)$ or $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j) = \infty$. Assume for contradiction that the counter values of $\mathtt{h}_j$ are not polynomially bounded.

Consider the case where $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j)$. Without loss of generality, we assume $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) < \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j)$. Therefore there exists a configuration pair $\langle (\mathbf{x}, p, m), (\mathbf{y}, q, n) \rangle$ in Π such that $m < |\mathcal{B}|$ and $\mathbf{x} \in \overline{\mathcal{W}}_1^{p,m}$. Using an argument similar to the one used to prove the pseudo-pumping lemma (Lemma 10), we show that we can pump out some portion from the sub-run inside the belt to reach a configuration pair in the background space with unequal distance and smaller counter values.

In the case where $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j) = \infty$, we can show that $\mathtt{c}_j \not\equiv_{|\mathcal{B}|} \mathtt{d}_j$. We can then apply Lemma 12 to show the existence of a shorter run, which enters the background space at a point $\mathtt{h}_{j'} = \langle \mathtt{c}_{j'}, \mathtt{d}_{j'} \rangle$ with smaller counter values such that $\mathtt{c}_{j'} \not\equiv_{|\mathcal{B}|} \mathtt{d}_{j'}$. ◀

Proof of Lemma 16. Consider the run Π. We have shown that the counter values of configuration pairs inside the belt space during this run and that of the first background point, if it exists, are polynomially bounded in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$. We also proved that the length of Π after the first background point is polynomially bounded. Since the counter values of configuration pairs inside the initial space are also polynomially bounded, we get that the maximum counter value in Π is polynomially bounded in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$. ◀

5 Non-deterministic ODCA

In this section, we consider the counter-determinacy restriction over weightless OCAs (equivalently, with weights from the boolean semiring). These results do not follow from previous sections, as booleans are not a field.

▶ **Example 18.** *The following languages are defined over the alphabet $\Sigma = \{a, b\}$ and are recognised by non-deterministic OCA with counter-determinacy.*
(a) *The language $\mathcal{L}_1 = \{a^n b a^n \mid n > 0\}$.*
(b) *The language $\mathcal{L}_2 = \{(a+b)^* \mid$ number of a's is greater than number of b's$\}$.*
(c) *The language $\mathcal{L}_3 = \{a^n (b+c)^m b (b+c)^k \mid m, n \in \mathbb{N}$ and $m > n\}$.*
Note that none of the above languages are definable by visibly pushdown automata.

We observe that the relationship between non-deterministic and deterministic ODCAs is similar to that between non-deterministic and deterministic finite automata. By definition, deterministic ODCAs have at most one unique path for any fixed word. Therefore, they are deterministic OCAs with counter-determinacy. It is also easy to observe that deterministic OCAs are deterministic ODCAs. It follows that deterministic ODCAs and deterministic OCAs are expressively equivalent. Similar to non-deterministic finite automata, we observe that non-deterministic ODCAs can be determinised by a subset construction of the states of the finite state machine. However, this results in an exponential blow-up. In Example 18, the deterministic ODCA that recognises the language $\mathcal{L}_3$ has to check whether every b encountered after reading the word $a^n(b+c)^{n+1}$ is at the kth position from the end. This will require at least $\mathcal{O}(2^k)$ states. On the other hand, there is a non-deterministic ODCA with $\mathcal{O}(k)$ states recognising the same language. Similar to finite automata, non-deterministic ODCAs are a "succinct" way to represent deterministic OCAs.

▶ **Theorem 19** (Determinisation)**.** *Given a non-deterministic ODCA, a polynomial space machine can output an equivalent deterministic ODCA of exponential size.*

The idea in proving the above theorem is a simple subset construction. The above result and the fact that equivalence of deterministic ODCA is in NL gives us the upper bound in the following theorem. The lower bound follows from that of NFAs [16].

▶ **Theorem 20.** *The equivalence problem for non-deterministic ODCA is* PSPACE*-complete.*

The equivalence of non-deterministic OCA is undecidable [24]. Our theorem shows that undecidability is due to non-determinism in the component that modifies the counter.

6 Conclusion

We introduced a new model called one-deterministic-counter automata. The model "separates" the machine into two components, (1) counter structure - that can modify the counter, and (2) finite state machine - that can access the counter. This separation of the "writing" and "reading" part gives some natural advantages to the model. We show that the equivalence

problem for weighted ODCA is in P if the weights are from a field while that of non-deterministic ODCA is in PSPACE. Note that the equivalence problems on weighted automata (where weights are from a field) and non-deterministic finite automata are in P and PSPACE respectively. On the other hand, the equivalence problem for non-deterministic OCA is undecidable and that of weighted OCA (weights from a field) is not-known. It will be interesting to look at other models where we can separate the "writing" and the "reading" parts. For example, a natural extension is to consider *stack-deterministic* pushdown automata - where a deterministic machine updates the stack. We also leave open the question of learning of weighted ODCAs.

References

1. Steven P. Abney, David A. McAllester, and Fernando Pereira. Relating probabilistic grammars and automata. In Robert Dale and Kenneth Ward Church, editors, *27th Annual Meeting of the Association for Computational Linguistics, University of Maryland, College Park, Maryland, USA, 20-26 June 1999*, pages 542–549. ACL, 1999. doi:10.3115/1034678.1034759.
2. Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211. ACM, 2004. doi:10.1145/1007352.1007390.
3. Stanislav Böhm and Stefan Göller. Language equivalence of deterministic real-time one-counter automata is NL-Complete. In Filip Murlak and Piotr Sankowski, editors, *MFCS*, volume 6907 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2011.
4. Stanislav Böhm, Stefan Göller, and Petr Jancar. Bisimilarity of one-counter processes is PSPACE-Complete. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2010.
5. Stanislav Böhm, Stefan Göller, and Petr Jancar. Equivalence of deterministic one-counter automata is NL-Complete. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 131–140. ACM, 2013. doi:10.1145/2488608.2488626.
6. Stanislav Böhm, Stefan Göller, and Petr Jancar. Bisimulation equivalence and regularity for real-time one-counter automata. *J. Comput. Syst. Sci*, 80(4):720–743, 2014.
7. Tomás Brázdil, Javier Esparza, Stefan Kiefer, and Antonín Kucera. Analyzing probabilistic pushdown automata. *Formal Methods Syst. Des*, 43(2):124–163, 2013.
8. Tomáš Brázdil, Antonín Kučera, and Oldřich Stražovský. On the decidability of temporal properties of probabilistic pushdown automata. In Volker Diekert and Bruno Durand, editors, *STACS 2005*, pages 145–157, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
9. Vojtech Forejt, Petr Jancar, Stefan Kiefer, and James Worrell. Bisimilarity of probabilistic pushdown automata. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 448–460. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.448.
10. Vojtech Forejt, Petr Jancar, Stefan Kiefer, and James Worrell. Language equivalence of probabilistic pushdown automata. *Inf. Comput*, 237:1–11, 2014.
11. Juraj Hromkovič and Georg Schnitger. On probabilistic pushdown automata. *Information and Computation*, 208(8):982–995, 2010. doi:10.1016/j.ic.2009.11.001.
12. Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. On the complexity of equivalence and minimisation for q-weighted automata. *Log. Methods Comput. Sci.*, 9(1), 2013. doi:10.2168/LMCS-9(1:8)2013.
13. Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ICALP '92, pages 101–112, Berlin, Heidelberg, 1992. Springer-Verlag.

14 Antonín Kucera, Javier Esparza, and Richard Mayr. Model checking probabilistic pushdown automata. *Log. Methods Comput. Sci.*, 2(1), 2006. doi:10.2168/LMCS-2(1:2)2006.

15 Antonín Kučera. Methods for quantitative analysis of probabilistic pushdown automata. *Electronic Notes in Theoretical Computer Science*, 149(1):3–15, 2006. Proceedings of the 7th International Workshop on Verification of Infinite-State Systems (INFINITY 2005). doi:10.1016/j.entcs.2005.11.013.

16 Meyer and Stockmeyer. Word problems requiring exponential time. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1973.

17 Dirk Nowotka and Jirí Srba. Height-deterministic pushdown automata. In Ludek Kucera and Antonín Kucera, editors, *MFCS*, volume 4708 of *Lecture Notes in Computer Science*, pages 125–134. Springer, 2007.

18 William Ogden. A helpful result for proving inherent ambiguity. *Mathematical Systems Theory*, 2(3):191–194, 1968.

19 Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Reasoning about recursive probabilistic programs. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pages 672–681, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2933575.2935317.

20 M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2–3):245–270, September 1961.

21 Géraud Sénizergues. The equivalence problem for deterministic pushdown automata is decidable. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming*, pages 671–681, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

22 Stirling. Deciding DPDA equivalence is primitive recursive. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2002.

23 Wen-Guey Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM J. Comput*, 21(2):216–227, 1992.

24 Valiant and Paterson. Deterministic one-counter automata. *JCSS: Journal of Computer and System Sciences*, 10, 1975.

A Section 3. Reachability problems of weighted ODCA

Given a weighted automaton $\mathcal{B}$ over a field $\mathcal{F}$, with k states, an initial configuration $\bar{\mathtt{c}}$, a vector space $\mathcal{U} \subseteq \mathcal{F}^k$ and a set of counter states S, the co-VS reachability problem asks whether there exists a run $\bar{\mathtt{c}} \xrightarrow{*} \overline{\mathcal{U}} \times S$.

▶ **Theorem 21.** *There is a polynomial time algorithm that decides the co-VS reachability problem for weighted automata and outputs a minimal reachability witness if it exists.*

Proof. The idea of equivalence checking of weighted automata goes back to the seminal paper by Schützenberger [20]. Tzeng [23] provided a polynomial time algorithm for the equivalence of two probabilistic automata by reducing the problem to the co-VS reachability problem where $\mathcal{V} = \{\mathbf{0}\}$. The same algorithm can be modified to solve the general co-VS reachability problem of weighted automata. ◀

▶ **Lemma 9.** *Let $z \in \Sigma^*$ be a minimal witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, X)$. If the number of distinct counter values encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times X$ is t, then $|z| \leq t \cdot K$.*

Proof. Let $\mathtt{c} = \mathtt{c}_1$ and $T(\mathtt{c}_1) = \mathtt{c}_1 \tau_1 \mathtt{c}_2 \cdots \tau_{h-1} \mathtt{c}_h$ be the run on word z from $\mathtt{c}_1$ and T the corresponding sequence of transitions. Let t be the number of distinct counter values encountered during this run. Now assume for contradiction that $h > |Q| \cdot |C| \cdot t$, then by Pigeonhole principle, there are $|Q| + 1$ many configurations $\mathtt{c}_{i_0}, \mathtt{c}_{i_1}, \ldots, \mathtt{c}_{i_{|Q|}}$ with the

same counter state and counter value during this run. Given a configuration $\mathtt{c}$, let $\mathbf{x}_{\mathtt{c}}$ denote WEIGHT-VECTOR($\mathtt{c}$). Let $\mathbb{A}_j$ denote the matrix such that $\mathbf{x}_{\mathtt{c}_{i_j}}\mathbb{A}_j = \mathbf{x}_{\mathtt{c}_h}$ for all $j \in [0, |Q|]$. Using linear algebra, we get that there exists $r \leq |Q|$, and $t \in [0, r-1]$ such that $\mathbf{x}_{\mathtt{c}_{i_t}}\mathbb{A}_r \in \overline{\mathcal{V}}$. Consider the sequence of transitions $T' = \tau_{1\cdots i_t}\tau_{r\cdots\ell-1}$ and $v = \mathtt{word}(T')$. The run $\pi(v, \mathtt{c}_1) = T'(\mathtt{c}_1)$ is a run since configurations $\mathtt{c}_t$ and $\mathtt{c}_r$ have the same counter state and counter value. This is a shorter run than $\pi(z, \mathtt{c}_1)$ and $\mathtt{c}_1 \xrightarrow{v} \overline{\mathcal{V}} \times S \times X$. This contradicts the minimality of z. ◀

▶ **Corollary 11.** *If $z \in \Sigma^*$ is a minimal witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$, then*

1. *the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is less than $max(\textsc{counter-value}(\mathtt{c}), m) + K^2$, and*
2. *$|z| \leq K^3 + max(\textsc{counter-value}(\mathtt{c}), m) \cdot K$.*

Proof. Let $z \in \Sigma^*$ be a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$, where $\mathtt{c}$ is a configuration with counter value n.

1. Consider the run of word z from $\mathtt{c}$. Let $\mathtt{d} \in \overline{\mathcal{V}} \times S \times \{m\}$ such that $\mathtt{c} \xrightarrow{z} \mathtt{d}$. Assume for contradiction that the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is greater than $max(n, m) + (|Q| \cdot |C|)^2$. Let $\mathtt{e}_1, \mathtt{e}_2, \cdots, \mathtt{e}_t$ be all the configurations in this run such that their counter values are zero. There exists words $u_1, u_2, \cdots, u_{t+1} \in \Sigma^*$ such that $z = u_1 u_2 \cdots u_{t+1}$ and $\mathtt{c} \xrightarrow{u_1} \mathtt{e}_1 \xrightarrow{u_2} \mathtt{e}_2 \xrightarrow{u_3} \cdots \xrightarrow{u_t} \mathtt{e}_t \xrightarrow{u_{t+1}} \mathtt{d}$. Note that $\mathtt{c} \xrightarrow{u_1} \mathtt{e}_1$, $\mathtt{e}_t \xrightarrow{u_{t+1}} \mathtt{d}$ and $\mathtt{e}_i \xrightarrow{u_{i+1}} \mathtt{e}_{i+1}$ for all $i \in [1, t-1]$ are floating runs (refer Figure 6).

We show that the counter values are bounded during these floating runs. First, we consider the floating run $\mathtt{c} \xrightarrow{u_1} \mathtt{e}_1$. Given a configuration $\mathtt{c}$, we use $\mathbf{x}_{\mathtt{c}}$ to denote WEIGHT-VECTOR($\mathtt{c}$). Let $\mathbb{A} \in \mathcal{F}^{|Q| \times |Q|}$ be such that $\mathbf{x}_{\mathtt{d}} = \mathbf{x}_{\mathtt{e}_1}\mathbb{A}$. The set $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{|Q|} \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}$ is a vector space and hence the vector $\mathbf{x}_{\mathtt{e}_1} \in \overline{\mathcal{U}}$. From Observation 8, we know that u_1 is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{U}}, \{p_{\mathtt{e}_1}\}, \{0\})$ and therefore by Lemma 10 we know that the maximum counter value encountered during the run $\pi(u_1, \mathtt{c})$ is less than $n + (|Q| \cdot |C|)^2$.

Similarly for the floating run $\mathtt{e}_t \xrightarrow{u_{t+1}} \mathtt{d}$, the maximum counter value is bounded by $m + (|Q| \cdot |C|)^2$. Now consider the floating runs $\mathtt{e}_i \xrightarrow{u_{i+1}} \mathtt{e}_{i+1}$ for all $i \in [1, t-1]$. Again by applying Lemma 10 we get that the maximum counter value encountered during these sub-runs is less than $(|Q| \cdot |C|)^2$. Therefore, the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is less than $max(n, m) + (|Q| \cdot |C|)^2$.

2. From the previous point, we know that the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is less than $max(n, m) + (|Q| \cdot |C|)^2$. Therefore, there are at most $max(n, m) + (|Q| \cdot |C|)^2$ many distinct counter values encountered during this run. Now from Lemma 9 we get that $|z| \leq (|Q| \cdot |C|) \cdot (max(n, m) + (|Q| \cdot |C|)^2)$. ◀

▶ **Lemma 12** (cut lemma). *Let $z \in \Sigma^*$ be a witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, where $\mathtt{c}$ is a configuration with $\textsc{counter-value}(\mathtt{c}) = n$ for some $n \in \mathbb{N}$, and $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run for some $m \in \mathbb{N}$. If $m - n > K$, then there exists $z_{sub} \in \Sigma^*$ such that z_{sub} is a subword of z, $\mathtt{c} \xrightarrow{z_{sub}} \overline{\mathcal{V}} \times S \times \{m'\}$ is a floating run and $m' - n < m - n$.*

Proof. Let $z \in \Sigma^*$ be a witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$ and $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run. Let n be the counter value of configuration $\mathtt{c}$ and $m > n + |Q| \cdot |C|$. Let $\mathtt{c}_1 = \mathtt{c}$ and $\pi(z, \mathtt{c}_1) = \mathtt{c}_1\tau_1\mathtt{c}_2 \cdots \tau_{\ell-1}\mathtt{c}_\ell$ be such that configuration $\mathtt{c}_\ell$ has counter value m. Consider the sequence of transitions $T = \tau_0\tau_1 \cdots \tau_{\ell-1}$ in $\pi(z, \mathtt{c}_1)$.

Since there are only $|C|$ counter states, by the Pigeonhole principle, there exists a strictly increasing sequence $I = 0 < i_0 < i_1 < \cdots < i_{|Q|} \leq \ell$ such that for all $j, j' \in I$ (Condition 1) $\textsc{counter-state}(\mathtt{c}_j) = \textsc{counter-state}(\mathtt{c}_{j'})$ and (Condition 2) if $j < j'$ then for all $d \in [j+1, j'-1]$, $\textsc{counter-value}(\mathtt{c}_j) < \textsc{counter-value}(\mathtt{c}_d) < \textsc{counter-value}(\mathtt{c}_{j'})$

and COUNTER-VALUE($\mathtt{c}_j$) < COUNTER-VALUE($\mathtt{c}_{j'}$). Given a configuration $\mathtt{c}$, let $\mathbf{x}_{\mathtt{c}}$ denote WEIGHT-VECTOR($\mathtt{c}$). Consider the set of configurations $\mathtt{c}_{i_0}, \mathtt{c}_{i_1}, \ldots, \mathtt{c}_{i_{|Q|}}$. For any $j \in [0, |Q|]$, let $\mathbb{A}_j$ denote the matrix such that $\mathbf{x}_{\mathtt{c}_{i_j}} \mathbb{A}_j = \mathbf{x}_{\mathtt{c}_\ell}$. Since $\mathbf{x}_{\mathtt{c}_{i_d}} \mathbb{A}_d \in \overline{\mathcal{V}}$ for all $d \in [0, |Q|]$, using linear algebra, we get that there exists $l, k \in [0, |Q|]$ with $l < k$ such that $\mathbf{x}_{\mathtt{c}_{i_l}} \mathbb{A}_k \in \overline{\mathcal{V}}$. Consider a configuration $\mathtt{e} = (\mathbf{x}, p, n)$. If $\pi(u, \mathtt{e})$ is a floating run with $min_{\mathtt{ce}}(\pi(u, \mathtt{e})) > 0$, then for all $m \in \mathbb{N}$ and $\mathbf{y} \in \mathcal{F}^{|Q|}$, $\pi(u, (\mathbf{y}, p, m))$ is a run. Consider the sequence of transitions $T' = \tau_{i_k \cdots \ell - 1}$ and let $u = \mathtt{word}(T')$. Because of Condition 2, $min_{\mathtt{ce}}(\pi(u, \mathtt{c}_{i_k})) > 0$. Therefore the run $T''(\mathtt{c}_1)$ where $T'' = \tau_{1 \cdots i_l - 1} \tau_{i_k \cdots \ell - 1}$ is a run shorter than $\pi(z, \mathtt{c}_1)$ with smaller counter effect. ◀

Now we show that for any run (need not be floating) of a minimal coverability witness z for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \mathbb{N}$ is polynomially bounded in the number of states of the machine and the initial counter value.

▶ **Corollary 22.** *If $z \in \Sigma^*$ is a minimal witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, where $\mathtt{c}$ is a configuration with counter value n, then the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \mathbb{N}$ is less than $max(n, |Q| \cdot |C|) + (|Q| \cdot |C|)^2$.*

Proof. Let $z \in \Sigma^*$ be a minimal witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, where $\mathtt{c}$ is a configuration with counter value n. Consider the run of word z from $\mathtt{c}$. Let $\mathtt{d} \in \overline{\mathcal{V}} \times S \times \mathbb{N}$ such that $\mathtt{c} \xrightarrow{z} \mathtt{d}$. If $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is a floating run, then by Lemma 12 the maximum counter value encountered during this run will be less than $n + |Q| \cdot |C|$. Now if $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is not a floating run, then there exists $u_1, u_2 \in \Sigma^*$ such that $z = u_1 u_2$ and $\mathtt{c} \xrightarrow{u_1} \mathtt{e} \xrightarrow{u_2} \mathtt{d}$ where, counter value of configuration $\mathtt{e}$ is zero and $\mathtt{e} \xrightarrow{u_2} \mathtt{d}$ is a floating run.

Given a configuration $\mathtt{c}$, let $\mathbf{x}_{\mathtt{c}}$ denote WEIGHT-VECTOR($\mathtt{c}$). Let $\mathbb{A} \in \mathcal{F}^{|Q| \times |Q|}$ be such that $\mathbf{x}_{\mathtt{d}} = \mathbf{x}_{\mathtt{e}} \mathbb{A}$. The set $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{|Q|} \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}$ is a vector space and hence the vector $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{U}}$. Note that for all $\mathbf{y} \in \overline{\mathcal{U}}$, the vector $\mathbf{y}\mathbb{A} \in \overline{\mathcal{V}}$. From Observation 8, we know that u_1 is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{U}}, \{p_{\mathtt{e}}\}, \{0\})$, where $p_{\mathtt{e}}$ is the counter state of configuration $\mathtt{e}$, and therefore by Corollary 11, we know that the maximum counter value encountered during the run $\pi(u_1, \mathtt{c})$ is less than $n + (|Q| \cdot |C|)^2$. Now since $\mathtt{e} \xrightarrow{u_2} \mathtt{d}$ is a floating run and u_2 is the minimal such word, from Lemma 12, we get that the counter value of configuration $\mathtt{d}$ is less than or equal to $|Q| \cdot |C|$, and by Lemma 10, we know that the maximum counter value encountered during this run is less than $|Q| \cdot |C| + (|Q| \cdot |C|)^2$. Therefore, we get that the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is less than $max(n, |Q| \cdot |C|) + (|Q| \cdot |C|)^2$. ◀

Our next objective is to show that the counter values are polynomially bounded during the run of a minimal coverability witness.

▶ **Corollary 23.** *Let $\mathtt{c}$ be a configuration with counter value n. If z is a minimal witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$ then $|z| \leq (|Q| \cdot |C|) \cdot (max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2)$.*

Proof. Let $z \in \Sigma^*$ be a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$. From Corollary 22, we know that the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \mathbb{N}$ is less than $max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2$. Therefore, there are at most $max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2$ many distinct counter values encountered during this run. Now from Lemma 9 we get that $|z| \leq (|Q| \cdot |C|) \cdot (max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2)$. ◀

▶ **Theorem 13.** *The co-VS reachability and co-VS coverability problems for weighted ODCA can be decided in polynomial time when the counter values are given in unary notation.*

Proof. Assume we are given a weighted ODCA $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}))$, initial configuration $\mathsf{c} = (\mathbf{x}, p, n)$, vector space $\mathcal{V}$, set of counter states S and counter value m as inputs for the co-VS reachability problem. For solving this reachability problem, we first consider the $max(n, m) + (|Q| \cdot |C|)^2$-unfolding weighted automaton $\mathcal{A}^{max(n,m)+(|Q|\cdot|C|)^2} = (C', \delta', p'_0;\ Q', \boldsymbol{\lambda}', \Delta', \boldsymbol{\eta}'_F)$ of $\mathcal{A}$. From Corollary 11, we know that the maximum counter value encountered during the run of the minimal reachability witness z for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$ is less than $max(n, m) + (|Q| \cdot |C|)^2$. We define a vector space $\mathcal{U} \subseteq \mathcal{F}^{|Q'|}$ as follows: A vector $\mathbf{z} \in \mathcal{F}^{|Q'|}$ is in $\mathcal{U}$ if there exists $\mathbf{y} \in \mathcal{V}$ such that for all $i \in [0, |Q| - 1]$, $\mathbf{z}[|Q| \cdot m + i] = \mathbf{y}[i]$ and for all $m' \neq m$ and $i \in [0, |Q| - 1]$, $\mathbf{z}[|Q| \cdot m' + i] = 0$.

Given a configuration $\mathsf{c} = (\mathbf{x}, p, n)$ of a weighted ODCA, we define the vector $\mathbf{z}_{\mathsf{c}} \in \mathcal{F}^{|Q'|}$.

$$\mathbf{z}_{\mathsf{c}}[i] = \begin{cases} \mathbf{x}[i \bmod |Q|], & \text{if } \frac{i}{|Q|} = n \\ 0, & \text{otherwise} \end{cases}$$

Now, consider the configuration $\bar{\mathsf{c}} = (\mathbf{z}_{\mathsf{c}}, (p, n))$ of $\mathcal{A}^{max(n,m)+(|Q|\cdot|C|)^2}$ and check whether $\bar{\mathsf{c}} \xrightarrow{*} \overline{\mathcal{U}} \times S \times \{0\}$. This is a co-VS reachability problem of weighted automata. Using Theorem 21, this can be solved in polynomial time.

For solving co-VS coverability problem when a weighted ODCA $\mathcal{A}$ with an initial configuration $\mathsf{c} = (\mathbf{z}, p, n)$, a vector space $\mathcal{V}$ and a set of counter states S are given as inputs, we consider the $max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2$-unfolding weighted automaton $\mathcal{A}^{max(n,(|Q|\cdot|C|))+(|Q|\cdot|C|)^2} = (C', \delta', p'_0;\ Q', \boldsymbol{\lambda}', \Delta', \boldsymbol{\eta}'_F)$ of $\mathcal{A}$. From Corollary 22, we know that the maximum counter value encountered during the run of a minimal reachability witness z for $(\mathsf{c}, \overline{\mathcal{V}}, S, \mathbb{N})$ is less than $max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2$. We define a vector space $\mathcal{U} \subseteq \mathcal{F}^{|Q'|}$ as follows: A vector $\mathbf{x} \in \mathcal{F}^{|Q'|}$ is in $\mathcal{U}$ if there exists $\mathbf{y} \in \mathcal{V}$ and $m \in \mathbb{N}$ such that for all $i \in [0, |Q| - 1]$, $\mathbf{x}[|Q| \cdot m + i] = \mathbf{y}[i]$ and for all $m' \neq m$ and $i \in [0, |Q| - 1]$, $\mathbf{x}[|Q| \cdot m' + i] = 0$. Given a configuration $\mathsf{c} = (\mathbf{x}, p, n)$ of a weighted ODCA, we define the vector $\mathbf{z}_{\mathsf{c}} \in \mathcal{F}^{|Q'|}$.

$$\mathbf{z}_{\mathsf{c}}[i] = \begin{cases} \mathbf{x}[i \bmod |Q|], & \text{if } \frac{i}{|Q|} = n \\ 0, & \text{otherwise} \end{cases}$$

Now, consider the configuration $\bar{\mathsf{c}} = (\mathbf{z}_{\mathsf{c}}, (p, n))$ of $\mathcal{A}^{max(n,(|Q|\cdot|C|))+(|Q|\cdot|C|)^2}$ and check whether $\bar{\mathsf{c}} \xrightarrow{*} \overline{\mathcal{U}} \times S \times \{0\}$. This is a co-VS reachability problem of a weighted automaton. From Theorem 21, we know that this can be solved in polynomial time. ◀

▶ **Lemma 15** (special-word lemma). *If $z \in \Sigma^*$ is the lexicographically minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, then there exists $u, y_1, v_1, v_2, v_3, y_2, w \in \Sigma^*$ and $r_1, r_2 \in \mathbb{N}$ such that $z = uy_1^{r_1}vy_2^{r_2}w$ and the following are true:*

1. *$|uy_1vy_2w|$ is polynomially bounded in the number of states of the machine.*
2. *r_1 and r_2 are polynomially bounded in the number of states of the machine, m, and* COUNTER-VALUE(c).

Proof. Let $z \in \Sigma^*$ be the lexicographically minimal reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, where c is a configuration with counter value n. Consider the run of word z from c. Let $\mathsf{d} \in \overline{\mathcal{V}} \times S \times \{m\}$ such that $\mathsf{c} \xrightarrow{z} \mathsf{d}$. Let $\mathsf{c} = \mathsf{c}_1$ and $T(\mathsf{c}_1) = \mathsf{c}_1\tau_1\mathsf{c}_2 \cdots \tau_{\ell-1}\mathsf{c}_\ell$ denote the run on word z from the configuration c_1 and T the corresponding sequence of transitions. Let e_1 be the first configuration with counter value zero and e_2 be the last configuration with counter value zero during this run. Let $z_1, z_2, z_3 \in \Sigma^*$ be such that $\mathsf{c} \xrightarrow{z_1} \mathsf{e}_1 \xrightarrow{z_2} \mathsf{e}_2 \xrightarrow{z_3} \mathsf{c}_\ell$ and $z = z_1z_2z_3$. Observe that $\mathsf{c} \xrightarrow{z_1} \mathsf{e}_1$ and $\mathsf{e}_2 \xrightarrow{z_3} \mathsf{c}_\ell$ are floating runs.

From Lemma 14, we know that there exists $u_1, u_3, v_1, v_3, y_1, y_3 \in \Sigma^*$ and $r_1, r_3 \in \mathbb{N}$ such that $z_1 = u_1 y_1^{r_1} v_1$, $z_3 = u_3 y_3^{r_3} v_3$, $|u_1 y_1 v_1 u_3 y_3 v_2| \leq 2 \cdot p_1(|Q| \cdot |C|)$, $r_1 \leq p_2(|Q| \cdot |C|, n)$, and $r_3 \leq p_2(|Q| \cdot |C|, m)$. Also, from Corollary 11 we get that $|z_2| \leq (|Q| \cdot |C|)^3$. ◀

B Regularity of ODCA is in P

We say that a weighted ODCA $\mathcal{A}$ is regular if there is a weighted automaton $\mathcal{B}$ that is equivalent to it. We look at the regularity problem - the problem of deciding whether a weighted ODCA is regular. We fix a weighted ODCA $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}))$ and use N to denote $|C| \cdot |Q|$. The proof technique is adapted from the ideas developed by Böhm et al. [6] in the context of real-time OCA. The crucial idea in proving regularity is to check for the existence of infinitely many equivalence classes. The proof relies on the notion of distance of configurations. Distance of a configuration is the length of a minimal word to be read to reach a configuration that does not have an N equivalent configuration in the underlying weighted automaton. The challenge is to find infinitely many configurations reachable from the initial configuration, so that no two of them have same distance.

▶ **Theorem 4.** *The regularity problem of weighted ODCA (weights from a field) is in* P.

Proof. Let $\mathcal{A}$ be a weighted ODCA and c_0 be its initial configuration. Lemma 24 shows that if $\mathcal{A}$ is not regular, then there are words $u, v \in \Sigma^*$ and configurations d, e such that there is a run of the form $\mathsf{c}_0 \xrightarrow{u} \mathsf{d} \xrightarrow{v} \mathsf{e}$ such that $\mathsf{N}^2 + \mathsf{N} \leq \text{COUNTER-VALUE}(\mathsf{d}) \leq 2\mathsf{N}^2 + \mathsf{N}$, $\text{WEIGHT-VECTOR}(\mathsf{e}) \in \overline{\mathcal{W}}^{\text{COUNTER-STATE}(\mathsf{e}),\text{COUNTER-VALUE}(\mathsf{e})}$ with $\text{COUNTER-VALUE}(\mathsf{e}) < \mathsf{N}$. The existence of such words u and v can be decided in polynomial time since the minimal length of such a path if it exists, is polynomially bounded in the number of states of the weighted ODCA by Corollary 11. This concludes the proof. ◀

We use c to denote some configuration of $\mathcal{A}$ and $\bar{\mathsf{c}}$ to denote some configuration of $\mathrm{U}(\mathcal{A})$. For a $p \in C$ and $m \in \mathbb{N}$, we define $\mathcal{W}^{p,m} = \{\mathbf{x} \in \mathcal{F}^{|Q|} \mid \exists \bar{\mathsf{c}} \in \mathcal{F}^{\mathsf{N}}, \mathsf{c} = (\mathbf{x}, p, m) \sim_{\mathsf{N}} \bar{\mathsf{c}}\}$. The set $\overline{\mathcal{W}}^{p,m}$ is $\mathcal{F}^{|Q|} \setminus \mathcal{W}^{p,m}$. The distance of a configuration c (denoted by $\text{dist}(\mathsf{c})$) is

$$\text{dist}(\mathsf{c}) = \min\{|w| \mid \mathsf{c} \xrightarrow{w} (\mathbf{x}, p, m)\ \exists p \in C, m < \mathsf{N}, \text{ and } \mathbf{x} \in \overline{\mathcal{W}}^{p,m}\}\ .$$

The following lemma shows when $\mathcal{A}$ is not regular. Given any configuration c, we use $\mathbf{x}_{\mathsf{c}}$ to denote $\text{WEIGHT-VECTOR}(\mathsf{c})$, p_{c} to denote $\text{COUNTER-STATE}(\mathsf{c})$ and n_{c} to denote $\text{COUNTER-VALUE}(\mathsf{c})$.

▶ **Lemma 24.** *Let* c *be the initial configuration of an ODCA* $\mathcal{A}$*. The following are equivalent.*

1. $\mathcal{A}$ *is not regular.*
2. *for all* $t \in \mathbb{N}$*, there exists configurations* d, e *s.t.* $n_{\mathsf{e}} < \mathsf{N}, \mathsf{c} \xrightarrow{*} \mathsf{d} \xrightarrow{*} \mathsf{e}$*,* $\mathbf{x}_{\mathsf{e}} \in \overline{\mathcal{W}}^{p_{\mathsf{e}}, n_{\mathsf{e}}}$ *and* $t < \text{dist}(\mathsf{d}) < \infty$*.*
3. *there exists configurations* d, e *and a run* $\mathsf{c} \xrightarrow{*} \mathsf{d} \xrightarrow{*} \mathsf{e}$ *s.t.* $\mathsf{N}^2 + \mathsf{N} \leq n_{\mathsf{d}} \leq 2\mathsf{N}^2 + \mathsf{N}$*,* $\mathbf{x}_{\mathsf{e}} \in \overline{\mathcal{W}}^{p_{\mathsf{e}}, n_{\mathsf{e}}}$ *with* $n_{\mathsf{e}} < \mathsf{N}$*.*

Proof. $3 \to 2$: Consider an arbitrary $q \in C$, $m < \mathsf{N}$ and vector space $\mathcal{V} = \mathcal{W}^{q,m}$. Let us assume for contradiction the complement of Point 2. That is, there exists a $t \in \mathbb{N}$ such that for all configurations d' where $\mathsf{c} \xrightarrow{*} \mathsf{d}' \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$, $\text{dist}(\mathsf{d}') \leq t$. Note that for all d' where $n_{\mathsf{d}'} > \mathsf{N}$, $\text{dist}(\mathsf{d}') \geq n_{\mathsf{d}'} - \mathsf{N}$. Hence there exists an $M \in \mathbb{N}$ such that for all d' where $\mathsf{c} \xrightarrow{*} \mathsf{d}' \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$, $n_{\mathsf{d}'} \leq M$.

Consider a configuration $\mathtt{d}$ where $n_{\mathtt{d}} > \mathsf{N}^2 + \mathsf{N}$ and a run $\mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$. Point 3 shows the existence of such a run. For contradiction, it suffices to show there exists a $\mathtt{d}'$ such that $\mathtt{c} \xrightarrow{*} \mathtt{d}' \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$ and $n_{\mathtt{d}'} > n_{\mathtt{d}}$. We get this from Lemma 10 Point 2, since $n_{\mathtt{c}} = 0$ and $n_{\mathtt{d}} > \mathsf{N}^2 + \mathsf{N}$.

$2 \to 1$: Assume for contradiction that for all $t \in \mathbb{N}$, there exists configurations $\mathtt{d}, \mathtt{e}$ such that $\mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \mathtt{e}$, $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}, n_{\mathtt{e}} < \mathsf{N}$ and $t < \mathrm{dist}(\mathtt{d}) < \infty$ but $\mathcal{A}$ is regular. Let $\mathcal{B}$ be the weighted automaton equivalent to $\mathcal{A}$. We use $|\mathcal{B}|$ to represent the number of states of $\mathcal{B}$.

Let $t_1, t_2, \ldots t_{|\mathcal{B}|+1} \in \mathbb{N}$ such that for $i \in [1, |\mathcal{B}|]$, $t_i < t_{i+1}$, and $\mathtt{d}_{t_i}$ be such that $\mathtt{c} \xrightarrow{*} \mathtt{d}_{t_i} \xrightarrow{*} (\mathbf{x}_i, p_{\mathtt{e}}, n_{\mathtt{e}})$, $\mathbf{x}_i \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$ and $t_i < \mathrm{dist}(\mathtt{d}_{t_i}) < t_{i+1} < \infty$. Clearly $\mathtt{d}_{t_i} \not\equiv \mathtt{d}_{t_j}$ for $i \neq j$ and corresponds to two different states of $\mathcal{B}$. Since we can find more than $|\mathcal{B}|$ pairwise non-equivalent configurations, it contradicts the assumption that $\mathcal{B}$ is equivalent to $\mathcal{A}$.

$1 \to 3$: We prove the contrapositive of the statement. Let us assume that there is no configurations $\mathtt{d}, \mathtt{e}$ and a run $\mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \mathtt{e}$ such that $\mathsf{N}^2 + \mathsf{N} \leq n_{\mathtt{d}} \leq 2\mathsf{N}^2 + \mathsf{N}$, $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$ with $n_{\mathtt{e}} < \mathsf{N}$. This implies that there does not exists a configuration $\mathtt{d}'$ such that $n_{\mathtt{d}'} > 2\mathsf{N}^2$, $\mathtt{c} \xrightarrow{*} \mathtt{d}' \xrightarrow{*} (\mathbf{y}, p_{\mathtt{e}}, n_{\mathtt{e}})$ for some $\mathbf{y} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$. Assume for contradiction that there is such a run, then there exists a portion in this run that can be "pumped down" to get a run $\mathtt{c} \xrightarrow{*} \mathtt{d}'' \xrightarrow{*} (\mathbf{y}', p_{\mathtt{e}}, n_{\mathtt{e}})$ for some configuration $\mathtt{d}''$ such that $\mathsf{N}^2 + \mathsf{N} \leq n_{\mathtt{d}''} \leq 2\mathsf{N}^2 + \mathsf{N}$ and $\mathbf{y}' \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$. This is a contradiction. Therefore all runs starting from configuration with counter value greater than or equal to $\mathsf{N}^2 + \mathsf{N}$ "looks" similar to a run on a weighted automaton. This allows us to simulate the runs of $\mathcal{A}$ using a weighted automaton. ◀

Comparing Infinitary Systems for Linear Logic with Fixed Points

Anupam Das
School of Computer Science, University of Birmingham, UK

Abhishek De
School of Computer Science, University of Birmingham, UK

Alexis Saurin
IRIF, CNRS, Université Paris Cité, France
INRIA π^3, Paris, France

Abstract

Extensions of Girard's *linear logic* by least and greatest fixed point operators (μMALL) have been an active field of research for almost two decades. Various proof systems are known *viz.* finitary and non-wellfounded, based on explicit and implicit (co)induction respectively. In this paper, we compare the relative expressivity, at the level of provability, of two complementary infinitary proof systems: finitely branching non-wellfounded proofs ($\mu\mathsf{MALL}^\infty$) vs. infinitely branching well-founded proofs ($\mu\mathsf{MALL}_{\omega,\infty}$). Our main result is that $\mu\mathsf{MALL}^\infty$ is strictly contained in $\mu\mathsf{MALL}_{\omega,\infty}$.

For inclusion, we devise a novel technique involving infinitary rewriting of non-wellfounded proofs that yields a wellfounded proof in the limit. For strictness of the inclusion, we improve previously known lower bounds on $\mu\mathsf{MALL}^\infty$ provability from Π^0_1-hard to Σ^1_1-hard, by encoding a sort of Büchi condition for Minsky machines.

2012 ACM Subject Classification Theory of computation → Linear logic; Theory of computation → Proof theory

Keywords and phrases linear logic, fixed points, non-wellfounded proofs, omega-branching proofs, analytical hierarchy

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.40

Funding The first and second authors are supported by a UKRI Future Leaders Fellowship, *Structure vs. Invariants in Proofs*, project reference MR/S035540/1. The third author is partially funded by the ANR project RECIPROG, project reference ANR-21-CE48-019-01.

1 Introduction

Fixed point logics have garnered significant interest from computational logicians over the years. In particular the extension of languages by least and fixed point operators, μ and ν respectively, has been comprehensively explored in modal logic [23, 38], arithmetic [22, 31], first-order logic [32, 14, 1], and linear logic [36, 7].

In terms of reasoning, least fixed points allow for *inductive* proof, while greatest fixed points, being dual to least fixed points, allow for *coinductive* proof. Naturally, the corresponding (co)induction proof rules must incorporate an arbitrary (co)invariant, a fundamental barrier to both proof theoretic investigations and (automated) proof search. To this end various alternative proof methods have been proposed, incorporating "infinitary behaviour" at the level of proofs:

- *Infinitary branching* (but wellfounded) systems have origins in the proof theory of arithmetic [10] and have been applied to numerous areas, including the modal μ-calculus [24, 37] and extensions of Kleene algebra [34, 27].
- More recently, *non-wellfounded* (but finitely branching) and *cyclic* proofs have been proposed for (co)inductive reasoning, originating in the modal μ-calculus [33, 2] and applied to theories of arithmetic [8, 9], type systems [26, 12, 11], and linear logic [36, 19, 6].

© Anupam Das, Abhishek De, and Alexis Saurin;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 40; pp. 40:1–40:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A natural question to ask is whether all these approaches prove the same theorems or not. In this work, we examine this question in the setting of linear logic, $\mu\mathsf{MALL}$. In particular, we compare the non-wellfounded system $\mu\mathsf{MALL}^\infty$ from [6] with a wellfounded infinitary branching system $\mu\mathsf{MALL}_{\omega,\infty}$ inspired by [16]. This builds on previous work [13] that focused on comparing the various *finitary* systems for $\mu\mathsf{MALL}$. Our main result is that $\mu\mathsf{MALL}_{\omega,\infty}$ proves strictly more theorems than $\mu\mathsf{MALL}^\infty$:

▶ **Theorem 1.** $\mu\mathsf{MALL}^\infty \subsetneq \mu\mathsf{MALL}_{\omega,\infty}$

Organisation and contributions. In Section 2, we recall the language of $\mu\mathsf{MALL}$ and present its various systems, in particular $\mu\mathsf{MALL}^\infty$ and $\mu\mathsf{MALL}_{\omega,\infty}$. In Section 3 we prove the inclusion part of Theorem 1. Namely, we give a coinductive translation from $\mu\mathsf{MALL}^\infty$ to $\mu\mathsf{MALL}_{\omega,\infty}$, and then exploit the correctness condition of $\mu\mathsf{MALL}^\infty$ to deduce that the image of this translation is wellfounded, Theorem 14. In Section 4 we reduce a "Büchi condition" for Minsky machines to $\mu\mathsf{MALL}^\infty$ provability, Proposition 32, implying the latter is Σ^1_1-hard by [3], Theorem 33, yielding the strictness part of Theorem 1. Finally in Section 5 we give a Π^1_2 upper bound for $\mu\mathsf{MALL}^\infty$, Theorem 35, by appealing to analytic determinacy of its "proof search game". We present conclusions in Section 6. Our results are summarised in Figure 1.

Notation. For a formula φ we write $\varphi^n(x)$ for $\overbrace{\varphi(\varphi(\cdots(\varphi}^{n}(x))\cdots)$. We shall also frequently suppress or explicitly indicate variables as convenient, e.g. we often identify φ and $\varphi(x)$, using the latter when we want to distinguish (some occurrences of) the variable x. When working with binders, e.g. μ and ν, we shall employ a standard convention of using dots, e.g. $\mu x.\varphi$ or $\nu x.\psi$ to signify that the μ or ν binds as far as possible to the right.

A note on (effective) descriptive set theory. In this work, we shall assume some familiarity with notions from (effective) descriptive set theory, namely the classes of the analytical hierarchy, Π^1_1, Σ^1_1, Π^1_2 etc. All necessary notions can be found in well-known textbooks like [29, 35] and via online resources.

2 Background

Linear logic, introduced by Girard [20], refines usual disjunction and conjunction into two orthogonal pairs of connectives: the *multiplicatives* $\parr, \otimes$ and the *additives* $\oplus, \&$. Together with their units $\bot, \mathbf{1}, \mathbf{0}, \top$ respectively, the resulting logic MALL ("multiplicative additive linear logic") is given in Figure 2 (colours may be ignored for now). Note here that the rules operate on *sequents*, which are finite multisets of formulas: commas denote multiset union, and set braces are omitted. Sequents are "one-sided", i.e. Γ should be read as $\vdash \Gamma$.

MALL is distinguished from usual logics by its notable absence of structural rules for the multiplicatives: $\varphi \parr \varphi \vdash \varphi$ and $\mathbf{0} \vdash \varphi$ are not always satisfied. This is why sequents must be multisets (or lists), not sets. In a sense linear logic can be seen as a "symmetrisation" of intuitionistic logic, which only controls structural rules on one side of an implication, resulting in a sort of constructive logic that nonetheless enjoys a form of De Morgan duality, hence admitting the one-sided presentation herein.

This lack of structural behaviour is crucially what leads to the high complexity of provability in the presence of "exponentials" in usual linear logic or, in this work, in the presence of fixed points. See [13, Sect. 2] for some further discussion on the peculiarities of linear logic with fixed points compared to other similar logics.

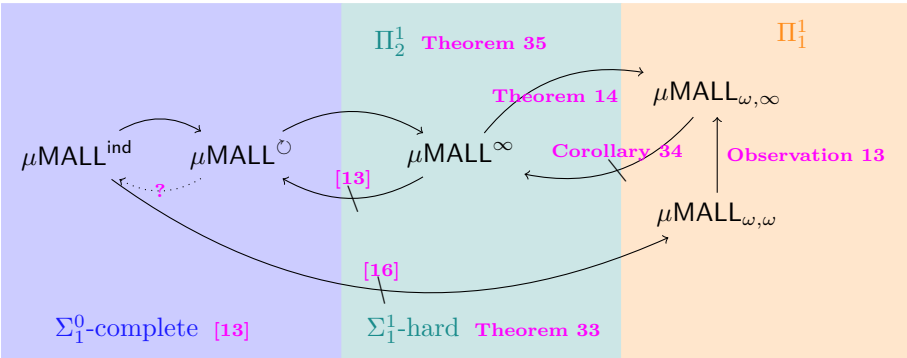


Figure 1 Relationships between systems in this work. Solid arrows $\rightarrow$ denote inclusion, dashed arrows denote conservative extensions, negated arrows $\not\rightarrow$ denote non-inclusion.

In the remainder of this section, we shall introduce the language of (multiplicative additive) linear logic with fixed points, and present the systems investigated in this work.

2.1 μMALL preliminaries

Let us fix two disjoint countable sets of propositional constants $\mathcal{A} = \{a, b, \dots\}$ and variables $\mathcal{V} = \{x, y, \dots\}$.

▶ **Definition 2** ((Pre)-formulas). μMALL **pre-formulas** *are given by the following grammar.*

$$\varphi, \psi ::= \mathbf{0} \mid \top \mid \bot \mid \mathbf{1} \mid a \mid a^\perp \mid x \mid \varphi \parr \psi \mid \varphi \otimes \psi \mid \varphi \oplus \psi \mid \varphi \& \psi \mid \mu x \varphi \mid \nu x \varphi$$

where $a \in \mathcal{A}$, $x \in \mathcal{V}$, and μ, ν bind the variable x in φ. Free and bound variables, and capture-avoiding substitution are defined as usual. The subformula ordering is denoted $\leq$. When a pre-formula is closed (i.e. has no free variable), we simply call it a **formula**.

$\mu x \varphi$ and $\nu x \varphi$ are intended to denote the least and greatest fixed points of the operator $\lambda x \varphi$ in an appropriate semantics (cf., e.g., [16]). $a^\perp$ is intended to be the negation of a. Note that, since variables have no negated instances, positivity of fixed point operators is implicit and no further condition is required.

Thanks to De Morgan duality in linear logic we may extend negation to all (pre-)formulas as a meta-operation, in the same way as for classical logic:

▶ **Definition 3.** **Negation** *of a pre-formula φ, denoted $\varphi^\perp$, is the unique involution that satisfies the following.*

$$(\mathbf{0})^\perp = \top; \quad (\bot)^\perp = \mathbf{1}; \quad a^{\perp\perp} = a; \quad x^\perp = x;$$
$$(\varphi \parr \psi)^\perp = \varphi^\perp \otimes \psi^\perp; \quad (\varphi \oplus \psi)^\perp = \varphi^\perp \& \psi^\perp; \quad (\mu x \varphi)^\perp = \nu x \varphi^\perp.$$

As expected, μ and ν are dual to each other; note also that fixed point variables are simply invariant under negation.

The first systems for μMALL, here called μMALL^{ind}, incorporate explicit (co)induction rules for the fixed points, inspired by similar developments in other fixed point logics like the μ-calculus [23, 39]. In our one-sided setting, μMALL^{ind} is formally the extension of the system MALL in Figure 2 by:

$$\frac{\Gamma, \varphi(\mu x \varphi)}{\Gamma, \mu x \varphi}\,(\mu) \qquad \frac{\psi^\perp, \varphi(\psi) \quad \Gamma, \psi}{\Gamma, \nu x \varphi}\,(\text{coind}) \tag{1}$$

Structural rules	$\dfrac{}{\varphi, \varphi^\perp}\,(\mathsf{id})$	$\dfrac{\Gamma_1, \varphi \quad \Gamma_2, \varphi^\perp}{\Gamma_1, \Gamma_2}\,(\mathsf{cut})$		
Logical rules	$\dfrac{\Gamma, \varphi_1, \varphi_2}{\Gamma, \varphi_1 \parr \varphi_2}\,(\parr)$	$\dfrac{\Gamma_1, \varphi_1 \quad \Gamma_2, \varphi_2}{\Gamma_1, \Gamma_2, \varphi_1 \otimes \varphi_2}\,(\otimes)$	$\dfrac{\Gamma, \varphi_i}{\Gamma, \varphi_1 \oplus \varphi_2}\,(\oplus_i)$	$\dfrac{\Gamma, \varphi_1 \quad \Gamma, \varphi_2}{\Gamma, \varphi_1 \& \varphi_2}\,(\&)$
Logical rules (units)	$\dfrac{}{1}\,(1)$	$\dfrac{\Gamma}{\Gamma, \perp}\,(\perp)$	$\dfrac{}{\Gamma, \top}\,(\top)$	No rule for $\mathbf{0}$

Figure 2 Inference rules for MALL, where $i \in \{1, 2\}$. Purple formulas in premiss(es) and conclusion are called ***auxiliary*** and ***principal*** respectively.

These rules are inspired by the second-order encoding of fixed points: $\nu x\varphi = \exists x((x \multimap \varphi) \otimes \varphi)$. Proofs of $\mu\mathsf{MALL}^{\mathsf{ind}}$ are defined as usual, but the system plays little role in this work; we present it only for context. At the level of their rules, the other systems considered in this work will only differ from $\mu\mathsf{MALL}^{\mathsf{ind}}$ in their ν-rules, using alternatives for (coind). All systems we consider will have the (μ) rule above (differing from the development in [16]).

2.2 Non-wellfounded system $\mu\mathsf{MALL}^\infty$

The standard "non-wellfounded" system for μMALL, here called $\mu\mathsf{MALL}^\infty$, was introduced in [6], building on earlier work for the fragment without multiplicatives [36, 19]. It is an adaptation of systems for the modal μ-calculus from [33, 37] to the setting of linear logic.

▶ **Definition 4** ($\mu\mathsf{MALL}^\infty$ pre-proofs). *The rules of $\mu\mathsf{MALL}^\infty$ extend* MALL *by:*

$$\frac{\Gamma, \varphi(\mu x\varphi)}{\Gamma, \mu x\varphi}\,(\mu) \qquad \frac{\Gamma, \varphi(\nu x\varphi)}{\Gamma, \nu x\varphi}\,(\nu) \tag{2}$$

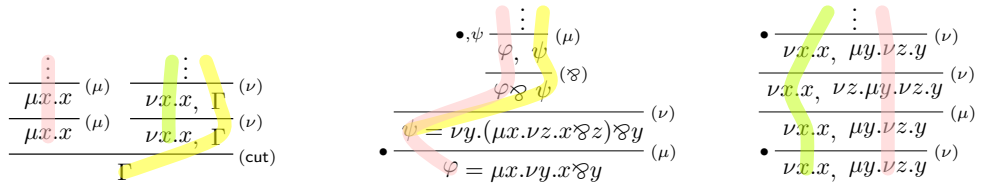
A **pre-proof** *of $\mu\mathsf{MALL}^\infty$, denoted $P, P', \ldots$, is a possibly non-wellfounded tree generated from the inference rules of $\mu\mathsf{MALL}^\infty$.*

Arbitrary non-wellfounded derivations may be fallacious, hence the affectation "pre-" above. Thus bona fide "proofs" must further satisfy a standard correctness criterion from non-wellfounded proof theory. At the same time the progressing criterion distinguishes the two fixed points, which have the same rules in Equation (2).

▶ **Definition 5** (Ancestry). *A formula occurrence φ in the conclusion of a rule instance is a* ***immediate ancestor*** *of an occurrence ψ in a premiss if they have the same colour, as typeset in Figure 2 and Equation (2). If φ and ψ are in a context $\Gamma, \Gamma_1, \Gamma_2$, we furthermore require that they are the same occurrences in the premiss and the conclusion.*

▶ Remark 6 (On occurrences in multisets). Note that, in the definition above, we are implicitly assuming that the data structure of a sequent allows us to distinguish different occurrences of the same formula. This is a standard convention in structural proof theory that avoids low-level peculiarities of working with lists (necessitating additional exchange/permutation rules). To be clear, "sequents-as-multisets" should be construed as a sets of occurences of formulas, e.g. by assigning a name to each occurrence. This is often made explicit in, e.g., type systems with explicit term annotations, but we gloss over this formality in favour of lightening the exposition.

▶ **Definition 7** (Threads and proof). *Given a branch B through a pre-proof, a* **thread** *is a maximal path in the graph of immediate ancestry of B. A thread is* **progressing** *if it has a minimal infinitely often principal formula, called* **progress points**, *(under $\leq$) that is a ν-formula. A pre-proof is a* ***proof*** *if each of its infinite branches has a progressing thread.*



(a) A pre-proof of arbitrary Γ. **(b)** ν-fairness $\not\Rightarrow$ progress. **(c)** A $\mu\mathsf{MALL}^\infty$ proof.

Figure 3 Some $\mu\mathsf{MALL}^\infty$ pre-proofs. We use identifiers like "$\bullet$" to describe infinite proofs in a finite manner. Progressing, non-progressing, and "stable" threads are indicated in green, red, and yellow respectively.

▶ **Example 8.** In Figure 3 we give several examples of (pre-)proofs. Figure 3a is a pre-proof of an arbitrary sequent Γ, exemplifying the inconsistency of arbitrary pre-proofs. It is *not* a proof because the left infinite branch has no progressing thread. Figure 3b is also not a proof, despite its only infinite branch having infinitely many (ν)-steps. This is because the thread indicated in red has the μ-formula φ as its minimal infinitely often principal formula, not the ν-formula ψ. Note that every other thread is eventually stable on ψ (and hence not progressing). Finally Figure 3c is indeed a $\mu\mathsf{MALL}^\infty$ proof, as its only infinite branch has a progressing thread on νxx. (It also happens to have a non-progressing red thread on $\mu y \nu z y$.)

In this work we shall make crucial use of a (nontrivial) cut-elimination result for $\mu\mathsf{MALL}^\infty$:

▶ **Theorem 9** ([6, 5]). *Every provable $\mu\mathsf{MALL}^\infty$ sequent has a proof without the* (cut) *rule.*

Finally, we briefly describe an important subsystem of $\mu\mathsf{MALL}^\infty$ where the underlying proof trees are *regular*.

▶ **Definition 10.** *A $\mu\mathsf{MALL}^\infty$ pre-proof is* **cyclic** *(a.k.a.* **regular***) if it has finitely many distinct sub-pre-proofs. $\mu\mathsf{MALL}^{\circlearrowleft}$ is the class of formulas with cyclic $\mu\mathsf{MALL}^\infty$.*

For instance the pre-proofs Figure 3a and Figure 3c are indeed regular whereas Figure 3b is not since at each iteration of the bullet the sequent has an extra occurrence of ψ (which is thenceforth non-principal). Like $\mu\mathsf{MALL}^{\mathsf{ind}}$, the circular system $\mu\mathsf{MALL}^{\circlearrowleft}$ will not play a significant role in this work.

▶ Remark 11 (On exponentials). For the reader familiar with the exponentials of linear logic, it would be reasonable to ask about the expressivity of extensions of $\mu\mathsf{MALL}^{\mathsf{ind}}, \mu\mathsf{MALL}^{\circlearrowleft}, \mu\mathsf{MALL}^\infty$ by the exponentials $!, ?$. It turns out that the resulting system is fully conservative over $\mu\mathsf{MALL}^{\mathsf{ind}}, \mu\mathsf{MALL}^{\circlearrowleft}, \mu\mathsf{MALL}^\infty$ respectively, thanks to the fact that exponentials can be "coded" by fixed point formulas, as noticed by Baelde in [4]. On the other hand, our previous work [13] showed the systems $\mu\mathsf{MALL}^{\mathsf{ind}}$ and $\mu\mathsf{MALL}^{\circlearrowleft}$ are already Σ^0_1-complete, matching the complexity of full linear logic with exponentials [28].

2.3 A well-founded system $\mu\mathsf{MALL}_{\omega,\infty}$

One of the main points of this work is to compare the non-wellfounded system $\mu\mathsf{MALL}^\infty$ with an orthogonal notion of infinite proof: well-founded but infinitely branching. Such systems are common in proof theory and mathematical logic [10, 30] and have been compared to non-wellfounded systems in other settings [37]. To this end, we consider an "ω-rule" for ν, motivated by continuous models, e.g. the phase semantics of [16].

▶ **Definition 12.** *$\mu\mathsf{MALL}_{\omega,\infty}$ is the extension of* MALL *by the rules:*

$$\frac{\Gamma, \varphi(\mu x\varphi)}{\Gamma, \mu x\varphi}\,(\mu) \qquad\qquad \frac{\Gamma, \top \quad \Gamma, \varphi(\top) \quad \Gamma, \varphi(\varphi(\top)) \quad \dots}{\Gamma, \nu x\varphi}\,(\omega) \tag{3}$$

Proofs of $\mu\mathsf{MALL}_{\omega,\infty}$ are defined as usual: they are well-founded (possibly infinite) trees generated by the rules of $\mu\mathsf{MALL}_{\omega,\infty}$.

The (ω) rule is inspired by the *inflationary* construction of fixed points, $\nu x\varphi = \bigcap_{\alpha\in\mathrm{Ord}} \varphi^\alpha(\top)$. It is implicit in $\mu\mathsf{MALL}_{\omega,\infty}$ that the ν operator is in a sense continuous, closing at ordinal ω, like in the models of phase semantics of [16]. In that work, a similar ω-branching system $\mu\mathsf{MALL}_{\omega,\omega}$ has been proposed for $\mu\mathsf{MALL}$ but it further restricts μ-rules to:

$$\frac{\Gamma, \varphi^n(\mathbf{0})}{\Gamma, \mu x\varphi}\,(\mu^n)$$

[16] shows that $\mu\mathsf{MALL}_{\omega,\omega}$ is actually quite weak and does not even contain $\mu\mathsf{MALL}^{\mathsf{ind}}$. Retaining the usual (μ) rule in $\mu\mathsf{MALL}_{\omega,\infty}$ is rather inspired by the *signatures* (a.k.a. *markings* or *assignments*) from [33, 11, 17]. In this work, we shall see that $\mu\mathsf{MALL}_{\omega,\infty}$ in fact contains all the systems we have presented. In particular, note that, since there is no rule for $\mathbf{0}$ in MALL, we immediately have:

▶ **Observation 13.** $\mu\mathsf{MALL}_{\omega,\omega} \subseteq \mu\mathsf{MALL}_{\omega,\infty}$

3 Inclusion of $\mu\mathsf{MALL}^\infty$ in $\mu\mathsf{MALL}_{\omega,\infty}$

In this section, we show one of our main results:

▶ **Theorem 14** (Simulating infinite height by infinite width)**.** $\mu\mathsf{MALL}^\infty \subseteq \mu\mathsf{MALL}_{\omega,\infty}$.

Note in particular the stark contrast with the system $\mu\mathsf{MALL}_{\omega,\omega}$ from [16], which does not even contain $\mu\mathsf{MALL}^{\mathsf{ind}}$, cf. Figure 1. To prove this result, throughout this section we work only with cut-free $\mu\mathsf{MALL}^\infty$ proofs, without loss of generality by Theorem 9. Furthermore, to prevent issues with productivity along a coinductive definition, we will employ a standard technique (e.g.[30]) of "bootstrapping" our $\mu\mathsf{MALL}$ systems with an explicit repetition rule $\frac{\Gamma}{\Gamma}\,(=)$.[1] While this does affect the notion of pre-proof it does not affect the notion of *proof* in $\mu\mathsf{MALL}^\infty$: the progressing condition implies that no infinite branch can have a tail of repetitions, and so $(=)$ steps can be contracted while preserving closedness (each sequent still concludes a step).

3.1 Projections

In order to prove Theorem 14 we first need to define notion of "proof projection" that allows us to extract from a $\mu\mathsf{MALL}^\infty$ proof its "finite approximations". Throughout this section we will consider sequents $\Gamma = \Gamma(\psi_1, \dots, \psi_k)$ where some occurrences of $\psi_1, \dots, \psi_k$ in Γ are distinguished. Note that the distinguished occurrences of, say ψ_i, may include some, none,

[1] Note that this addition could be avoided by using "explicit" approximants à la [37, 16].

or all of the occurrences of ψ_i in Γ. This notation allows for distinguished ψ_i occurrences to be subformulas of formulas in Γ, and also for some ψ_i and ψ_j to be the same formula when $i \neq j$. For $\vec{\psi} = (\nu x_1\varphi_1, \dots, \nu x_k\varphi_k)$, an **assignment** is simply a list $\vec{n} = (n_1, \dots, n_k) \in \omega^k$. We will write $\vec{\psi}^{\vec{n}} := (\varphi_1^{n_1}(\top), \dots, \varphi_k^{n_k}(\top))$, the list obtained by *assigning* each n_i to each ψ_i.

▶ **Definition 15** (Projections). *For a pre-proof P of $\Gamma(\vec{\psi})$, where $\vec{\psi} = (\nu x_1\varphi_1, \dots, \nu x_k\varphi_k)$, and an assignment $\vec{n} = (n_1, \dots, n_k) \in \omega^k$, we define the* **projection** *$P(\vec{n})$ a pre-proof of $\Gamma(\vec{\psi}^{\vec{n}})$ by coinduction on P as follows:*

1. *If P ends with a step ρ for which no distinguished formula occurrence is principal,*

$$\dfrac{\begin{array}{c}P_1\\ \Gamma_1(\vec{\psi})\end{array} \quad \cdots \quad \begin{array}{c}P_m\\ \Gamma_m(\vec{\psi})\end{array}}{\Gamma(\vec{\psi})}\,(\rho) \quad \text{, then} \quad P(\vec{n}) := \dfrac{\begin{array}{c}P_1(\vec{n})\\ \Gamma_1(\vec{\psi}^{\vec{n}})\end{array} \quad \cdots \quad \begin{array}{c}P_m(\vec{n})\\ \Gamma_m(\vec{\psi}^{\vec{n}})\end{array}}{\Gamma(\vec{\psi}^{\vec{n}})}\,(\rho).$$

2. *If P ends with a step for which some distinguished formula occurrence is principal,*

$$\dfrac{\begin{array}{c}P'\\ \varphi(\nu x\varphi), \Gamma(\nu x\varphi, \vec{\psi})\end{array}}{\nu x\varphi, \Gamma(\nu x\varphi, \vec{\psi})}\,(\nu) \quad \text{, then} \quad \begin{array}{l} P(0, \vec{n}) := \dfrac{}{\top, \Gamma(\top, \vec{\psi}^{\vec{n}})}\,(\top) \\ P(n+1, \vec{n}) := \dfrac{\begin{array}{c}P'(n, n+1, \vec{n})\\ \varphi(\varphi^n(\top)), \Gamma(\nu x\varphi, \vec{\psi})\end{array}}{\varphi^{n+1}(\top), \Gamma(\varphi^{n+1}(\top), \vec{\psi}^{\vec{n}})}\,(=) \end{array}.$$

Note that, in the final case of the definition above, the length of the assignment may increase if $\nu x\varphi$ distinguishes multiple occurrences in the sequent. This is why, even though we shall only ever use projections on a single formula later, we must make the definition above more general. This is also a barrier towards any arguments by explicit induction on assignments; e.g. Lemma 16 later is demonstrated rather by an argument by infinite descent, a now standard leitmotif of non-wellfounded proof theory.

3.2 Properties of branches along projections

For μMALL$^\infty$ pre-proofs P we associate to each of its (maximal) branches B its **induced** branch $B(\vec{n})$ in $P(\vec{n})$ in the expected way. Formally $B(\vec{n})$ is defined by coinduction on B, following the cases of Definition 15:

1. $$\left(\dfrac{\begin{array}{c}B_i\\ \Gamma_i(\vec{\psi})\end{array}}{\Gamma(\vec{\psi})}\,(\rho)\right)(\vec{n}) := \dfrac{\begin{array}{c}B_i(\vec{n})\\ \Gamma_i(\vec{\psi}^{\vec{n}})\end{array}}{\Gamma(\vec{\psi}^{\vec{n}})}\,(\rho)$$

2. $$\left(\dfrac{\begin{array}{c}B'\\ \varphi(\nu x\varphi), \Gamma(\nu x\varphi, \vec{\psi})\end{array}}{\nu x\varphi, \Gamma(\nu x\varphi, \vec{\psi})}\,(\nu)\right)(0, \vec{n}) := \dfrac{}{\top, \Gamma(\top, \vec{\psi}^{\vec{n}})}\,(\top)$$
$$\left(\dfrac{\begin{array}{c}B'\\ \varphi(\nu x\varphi), \Gamma(\nu x\varphi, \vec{\psi})\end{array}}{\nu x\varphi, \Gamma(\nu x\varphi, \vec{\psi})}\,(\nu)\right)(n+1, \vec{n}) := \dfrac{\begin{array}{c}B'(n, n+1, \vec{n})\\ \varphi(\varphi^n(\top)), \Gamma(\varphi^{n+1}(\top), \vec{\psi}^{\vec{n}})\end{array}}{\varphi^{n+1}(\top), \Gamma(\varphi^{n+1}(\top), \vec{\psi}^{\vec{n}})}\,(=)$$

Clearly the map $B \mapsto B(\vec{n})$ from branches of P to branches of $P(\vec{n})$ is surjective. It is also clear that if B is finite then so is $B(\vec{n})$. The remainder of this section is devoted to establishing a stronger property: as long as B is finite or progressing, so is $B(\vec{n})$. To this end we need the following important properties of the action of projections on threads:

▶ **Lemma 16** (Projections on progressing threads terminate). *For a $\mu\mathsf{MALL}^\infty$ pre-proof P of $\Gamma(\nu x\varphi, \vec{\psi})$, a branch B of P along which $\nu x\varphi$ extends to a progressing thread, and $n \in \omega$, the branch $B(n, \vec{n})$ is finite.*

Proof sketch. Suppose otherwise and take the (maximal) sequence $(n_i)_{i<\alpha\leq\omega}$ of numbers assigned to the progressing thread $\nu x\varphi$ in the construction of $B(n, \vec{n})$ above. By local inspection notice that $(n_i)_{i<\alpha}$ is monotone non-increasing, and furthermore strictly decreases whenever $\nu x\varphi$ is principal. Thus α must be finite and bounds the length of $B(n, \vec{n})$. ◀

We also have that projections "lower threads" disjoint from their distinguished formulas, by inspection of the description of $B(\vec{n})$ above:

▶ **Lemma 17** (Projections preserve disjoint threads). *Let P be a pre-proof of $\Gamma(\vec{\psi})$ and B a branch of P with $B(\vec{n})$ infinite. If B is progressing then so is $B(\vec{n})$. Moreover, if $(\varphi_i)_{i<\omega}$ is a progressing thread*[2] *along B* disjoint *from all $\vec{\psi}$ with progress points $(\varphi_{i_j})_{j<\omega}$, then $(\varphi_i)_{i<\omega}$ is also progressing in $B(\vec{n})$ with progress points $(\varphi_{i_j})_{j<\omega}$.*

Note that $B(\vec{n})$ may still be finite when B is infinite in case there is another progressing thread along B on a distinguished formula, cf. Lemma 16. Recalling that the map $B \mapsto B(\vec{n})$ from branches of P to branches of $P(\vec{n})$ is surjective, we have immediately from Lemma 17:

▶ **Proposition 18** (Projections on proofs are proofs). *If P is a $\mu\mathsf{MALL}^\infty$ proof, so is $P(\vec{n})$.*

3.3 The ω-translation

W e need to give a translation from $\mu\mathsf{MALL}^\infty$ proofs to $\mu\mathsf{MALL}_{\omega,\infty}$ ones. We break this up into two steps: first we give the translation, and then prove that the image of this translation is wellfounded. To this end we shall refer to "pre-proofs" of $\mu\mathsf{MALL}_{\omega,\infty}$ too, which may be both infinitely wide and infinitely deep.

▶ **Definition 19** (ω-translation). *For $\mu\mathsf{MALL}^\infty$ pre-proofs P, we define the $\mu\mathsf{MALL}_{\omega,\infty}$ pre-proof P^ω by coinduction on P as follows:*

1. *if* $P = \dfrac{\overset{P_1}{\Gamma_1} \quad \cdots \quad \overset{P_k}{\Gamma_k}}{\Gamma}\,(\rho)$ *with $\rho \neq \nu$, then* $P^\omega := \dfrac{\overset{P_1^\omega}{\Gamma_1} \quad \cdots \quad \overset{P_k^\omega}{\Gamma_k}}{\Gamma}\,(\rho)$

   (i.e. $\cdot^\omega$ commutes with ρ when $\rho \neq \nu$).

2. *Otherwise, if* $P = \dfrac{\overset{P'}{\Gamma, \varphi(\nu x\varphi)}}{\Gamma, \nu x\varphi}\,(\nu)$ *then* $P^\omega := \dfrac{\dfrac{}{\Gamma, \top}\,(\top) \quad \overset{P'(0)^\omega}{\Gamma, \varphi(\top)} \quad \overset{P'(1)^\omega}{\Gamma, \varphi(\varphi(\top))} \quad \cdots}{\Gamma, \nu x\varphi}\,(\omega)$.

Note that, whichever rule P ends with, the translation above is *productive* (it prints a rule for each coinductive case) and so P^ω is indeed well-defined by coinduction (just like projections and induced branches before). Note also that the translation is defined for arbitrary pre-proofs, not only proofs. Indeed a pre-proof P may be sent to a non-wellfounded pre-proof P^ω by the translation, e.g. if P has no (ν) step, then already $P^\omega = P$. In particular, simply having infinitely many (ν) steps along every infinite branch of P does not suffice to imply wellfoundedness of P^ω. Let us see some examples to illustrate this:

[2] Recall that P is cut-free, so we may assume the thread starts at the root.

▶ **Example 20** (ν-fairness $\not\Rightarrow$ wellfoundedness of $\cdot^\omega$)**.** Consider the $\mu\mathsf{MALL}^\infty$ pre-proof in Figure 3b. Recall that this pre-proof is not regular. This irregularity manifests in each branch of its image under the ω translation:

$$\dfrac{\dfrac{}{\top}\,(\top) \qquad \dfrac{\dfrac{\vdots}{\bullet, \varphi, \top}\,(\mu)}{\bullet, \varphi \parr \top}\,(\parr) \qquad \dfrac{\dfrac{\vdots}{\bullet, \varphi \parr \top, \varphi \parr \top}\,(\mu)}{\bullet, \varphi \parr (\varphi \parr \top)}\,(\parr) \qquad \cdots}{\bullet\, \dfrac{\psi}{\varphi}\,(\mu)}\,(\omega)$$

▶ **Example 21.** Consider the $\mu\mathsf{MALL}^\infty$ proof in Figure 3c. To compute its ω-translation let us first note that:

- When $\varphi(x) = x$ we have that $\varphi^n(\top) = \top$ for all $n < \omega$.
- When $\varphi(z) = \mu y.\nu z.y$ we have $\varphi^n(\top) = \mu y.\nu z.y$ for all $n < \omega$.

From here we can readily compute the ω-translation of Figure 3c as:

$$\dfrac{\dfrac{}{\top, \mu y.\nu z.y}\,(\top) \qquad \dfrac{\dfrac{\left\{\dfrac{}{\top, \mu y.\nu z.y}\,(\top)\right\}^\omega}{\top, \nu z.\mu y.\nu z.y}\,(\omega)}{\top, \mu y.\nu z.y}\,(\mu) \qquad \dfrac{\dfrac{\left\{\dfrac{\left\{\dfrac{}{\top, \mu y.\nu z.y}\,(\top)\right\}^\omega}{\top, \nu z.\mu y.\nu z.y}\,(\omega)\right\}^\omega}{\top, \mu y.\nu z.y}\,(\mu)}{\top, \nu z.\mu y.\nu z.y}\,(\omega) \qquad \cdots}{\nu x.x, \mu y.\nu z.y}\,(\omega)$$

3.4 Finiteness of branches in the image of the ω-translation

The above examples notwithstanding, we will indeed show that, as long as P is progressing, P^ω is actually wellfounded, and so is a $\mu\mathsf{MALL}_{\omega,\infty}$ proof after all. First we shall classify branches in the image of the ω-translation, just like we did for projections. Note that every branch of P^ω is induced from a branch of P by choosing, at each ν-step, a corresponding projection given by some $n \in \omega$. Thus, we may specify an arbitrary (possibly non-maximal) branch of P^ω by the notation $B^{\vec{n}}$, where B is a branch of P and $\vec{n} \in \omega^{\leq\omega}$ is some unique (possibly infinite) list of natural numbers, indexing the premisses of ω-steps followed by the branch. Formally $B^{\vec{n}}$ is defined by coinduction on B, following Definition 19, with a case analysis on the head of $\vec{n}$ in the case of a (ν) step:

$$\left(\dfrac{\begin{matrix} B_i \\ \Gamma_i \end{matrix}}{\Gamma}\,(\rho)\right)^{\vec{n}} := \dfrac{\begin{matrix} B_i^{\vec{n}} \\ \Gamma_i \end{matrix}}{\Gamma}\,(\rho) \quad (4)$$

$$\left(\dfrac{\begin{matrix} B' \\ \Gamma, \varphi(\nu x \varphi) \end{matrix}}{\Gamma, \nu x \varphi}\,(\nu)\right)^{\varepsilon} := \Gamma, \nu x \varphi$$

$$\left(\dfrac{\begin{matrix} B' \\ \Gamma, \varphi(\nu x \varphi) \end{matrix}}{\Gamma, \nu x \varphi}\,(\nu)\right)^{0\vec{n}} := \dfrac{\dfrac{}{\Gamma, \top}\,(\top)}{\Gamma, \nu x \varphi}\,(\omega) \quad (5)$$

$$\left(\dfrac{\begin{matrix} B' \\ \Gamma, \varphi(\nu x \varphi) \end{matrix}}{\Gamma, \nu x \varphi}\,(\nu)\right)^{(n+1)\vec{n}} := \dfrac{\begin{matrix} B'(n)^{\vec{n}} \\ \Gamma, \varphi^{n+1}(\top) \end{matrix}}{\Gamma, \nu x \varphi}\,(\omega)$$

▶ **Observation 22.** *If B is finite, then so is $B^{\vec{n}}$.*

This follows by induction on the length of B. From here we are able to show:

▶ **Lemma 23.** *For P a pre-proof, $B^{\vec{n}}$ a branch of P^ω: if B is progressing then $B^{\vec{n}}$ is finite.*

Formally this follows by induction on the height of the first progress point of a progressing thread along B, following the definition of branches $B^{\vec{n}}$. During the argument we must often appeal to the properties of branches along projections from Section 3.2. Of course from here our main result immediately follows:

Proof of Theorem 14. Let P be a $\mu\mathsf{MALL}^\infty$ proof. By Lemma 23 above, all branches of its ω-translation P^ω are finite. Thus P^ω is indeed wellfounded and so a proof of $\mu\mathsf{MALL}_{\omega,\infty}$. ◀

4 $\mu\mathsf{MALL}^\infty$ is Σ^1_1-hard

A natural question to ask now is if $\mu\mathsf{MALL}_{\omega,\infty}$ can be embedded in $\mu\mathsf{MALL}^\infty$. [37] shows that the ω-branching calculus of the modal μ-calculus can be embedded in its corresponding non-wellfounded calculus. The argument crucially depends on the fact that any proof of a formula φ has finitely many distinct sequents (modulo identifying approximations); however, such a condition does not hold in $\mu\mathsf{MALL}$ due to the absence of structural rules. In fact, we prove that the inclusion result of the previous section, Theorem 14, is strict.

In order to do so we will give a Σ^1_1 lower bound for $\mu\mathsf{MALL}^\infty$ that is incompatible with the natural Π^1_1 upper bound for $\mu\mathsf{MALL}_{\omega,\infty}$. To this end, we encode a Büchi' condition for Minsky machines in terms of $\mu\mathsf{MALL}^\infty$ provability. This significantly improves a Π^0_1 lower bound from previous work [13], which was proved by reduction from non-halting of Minsky machines.

Throughout this section we shall write a^n for $\overbrace{a \parr \ldots \parr a}^{n}$ (which is equivalent to $a^n(\bot)$).

▶ **Definition 24.** *A* **Minsky machine** *$\mathcal{M}$ is a tuple (Q, r_1, r_2, I) where Q is a finite set of states, r_1, r_2 are two registers and I is a set of* instructions *of the form $\mathsf{INC}(p, r_i, q)$ or $\mathsf{JZDEC}(p, r_i, q_0, q_1)$, for $p, q, q_0, q_1 \in Q$ and $i \in \{1, 2\}$, that manipulate the current state and the contents of the registers.*

The operational semantics of $\mathcal{M}$ is given by its configuration graph, whose vertices are **configurations**, *of form $\langle q, a, b\rangle \in Q \times \mathbb{N} \times \mathbb{N}$, and whose edges are induced from I by:*

$$\langle p, a, b\rangle \xrightarrow{\mathsf{INC}(p,r_1,q)} \langle q, a+1, b\rangle \qquad \langle p, a, b\rangle \xrightarrow{\mathsf{INC}(p,r_2,q)} \langle q, a, b+1\rangle$$

$$\langle p, 0, b\rangle \xrightarrow{\mathsf{JZDEC}(p,r_1,q_0,q_1)} \langle q_0, 0, b\rangle \qquad \langle p, a, 0\rangle \xrightarrow{\mathsf{JZDEC}(p,r_2,q_0,q_1)} \langle q_0, a, 0\rangle$$

$$\langle p, a+1, b\rangle \xrightarrow{\mathsf{JZDEC}(p,r_1,q_0,q_1)} \langle q_1, a, b\rangle \qquad \langle p, a, b+1\rangle \xrightarrow{\mathsf{JZDEC}(p,r_2,q_0,q_1)} \langle q_1, a, b\rangle$$

A **run** *is a maximal path in the configuration graph.*

▶ **Theorem 25** ([3]). *Given a Minsky machine $\mathcal{M}$ and a state q_0, checking whether* there exists *an infinite run starting from $\langle q_0, 0, 0\rangle$ that visits q_0 infinitely often is Σ^1_1-hard.*

For the rest of the section, let us fix a Minsky machine $\mathcal{M} = (Q, r_1, r_2, I)$. Construe $\{a, b, z_a, z_b\} \cup Q$ as a set of propositional constants (assuming $\{a, b, z_a, z_b\} \cap Q = \emptyset$) and $\{x, y\}$ as a set of variables. We use a and z_a (respectively, b and z_b) to represent the contents of the register r_1 (respectively, r_2). Define $\mathsf{parity} : Q \to \{x, y\}$ by $\mathsf{parity}(q) = x$ if $q = q_0$ and $\mathsf{parity}(q) = y$ otherwise. Define the following:

$$
\begin{aligned}
[\mathsf{INC}(p, r_1, q)] &:= p^\perp \otimes (q \parr a \parr \mathsf{parity}(q)) \\
[\mathsf{INC}(p, r_2, q)] &:= p^\perp \otimes (q \parr b \parr \mathsf{parity}(q)) \\
[\mathsf{JZDEC}(p, r_1, q, q')] &:= p^\perp \otimes (((\mathsf{parity}(q) \parr q) \& z_a) \oplus (a^\perp \otimes (\mathsf{parity}(q') \parr q'))) \\
[\mathsf{JZDEC}(p, r_2, q, q')] &:= p^\perp \otimes (((\mathsf{parity}(q) \parr q) \& z_b) \oplus (b^\perp \otimes (\mathsf{parity}(q') \parr q'))) \\
\psi &:= \mu y. \left(\bigoplus_{\mathsf{ins} \in I} [\mathsf{ins}] \right) \\
\varphi &:= \psi(\nu x.\psi)
\end{aligned}
$$

Finally, define $\mathsf{Inv} := ((b^\perp)^* \otimes z_a^\perp) \oplus ((a^\perp)^* \otimes z_b^\perp)$ where we write $\varphi^* = \mu x.(1 \oplus (\varphi \otimes x))$.

▶ **Proposition 26.** *For any $n \in \mathbb{N}$, the sequents b^n, z_a, Inv and a^n, z_b, Inv are provable.*

Define $\mathsf{CP} : Q \to \{\nu x.\psi, \varphi\}$ such that $\mathsf{CP}(q) = \nu x.\psi$ if $q = q_0$ and $\mathsf{CP}(q) = \varphi$ otherwise.

▶ **Lemma 27** (One step simulation)**.** *Let $\langle p, m, n\rangle$ be a configuration such that $\langle p, m, n\rangle \xrightarrow{\mathsf{ins}} \langle q, m', n'\rangle$, for $\mathsf{ins} \in I$. The following "move" gadget has a finite $\mu\mathsf{MALL}^\infty$ derivation:*

$$
\frac{\mathsf{CP}(q), q, a^{m'}, b^{n'}, \mathsf{Inv}}{\mathsf{CP}(p), p, a^m, b^n, \mathsf{Inv}} \, (mv_{\mathsf{ins}})
$$

Moreover, if $p = q_0$ then (mv_{ins}) has a (ν) step (for which $\nu x\psi$ is principal, necessarily).

▶ **Lemma 28.** *If there exists a run of $\mathcal{M}$ from q_0 such that q_0 is visited infinitely often, the sequent $\nu x.\psi, q_0, \mathsf{Inv}$ has a $\mu\mathsf{MALL}^\infty$ proof.*

Proof sketch. Let $\mathcal{R}(p_0) = (\langle p_i, m_i, n_i\rangle)_{0 \le i < \omega}$ be an infinite run of $\mathcal{M}$ from q_0 (so $p_0 = q_0$). We construct a pre-proof $P(p_0)$ of $\nu x.\psi, q_0, \mathsf{Inv}$ by coinduction on $\mathcal{R}(q_0)$, simply by simulating each step of the run by the one-step "move" gadgets from Lemma 27 (see Figure 4 for a visualisation). We now argue that $P(p_0)$ is progressing, and so is indeed a $\mu\mathsf{MALL}^\infty$ proof.

First, observe that $P(p_0)$ has exactly one infinite branch that has infinitely many occurrences of "move" gadgets (mv_{ins}). Furthermore, every time there is a move rule with a conclusion of the form $\mathsf{CP}(q_0), q_0, a^m, b^n, \mathsf{Inv}$, there is a (ν) step, necessarily on $\mathsf{CP}(q_0) = \nu x\psi$, by Lemma 27. So, since q_0 occurs infinitely often in the run, and by cut-freeness, there is an infinite thread τ along the formulas $\mathsf{CP}(p_i)$ which is infinitely often principal for $\mathsf{CP}(q_0) = \nu x\psi$ (the indicated green thread in Figure 4). Finally, by inspection of the formulas $\mathsf{CP}(p_i)$ and the rules of $\mu\mathsf{MALL}^\infty$, every formula occurring in τ must have $\nu x\psi$ as a subformula. Thus τ is indeed progressing, and so $P(p_0)$ is a $\mu\mathsf{MALL}^\infty$ proof as required. ◀

4.1 Background on focusing

In order to prove the converse of Lemma 28 above, we have to account for all possible proofs. In order to tame the space of possibilities we shall appeal to "focusing", a standard technique in proof search. Informally, focused proofs are a family of proofs that have more structure than usual sequent calculus proofs.

We first classify the connectives of $\mu\mathsf{MALL}$ by two polarities: **positive** and **negative**. Inferences for negative connectives are *invertible*, i.e. they preserve provability bottom-up, but the positive inferences do not in general. The negative (respectively, positive) connectives of $\mu\mathsf{MALL}^\infty$ are $\&, \parr, \perp, \top, \nu$ (respectively, $\otimes, \oplus, 1, 0, \mu$).[3]

[3] Observe that both the μ and ν rules are invertible. See [6] for an explanation of the choice.

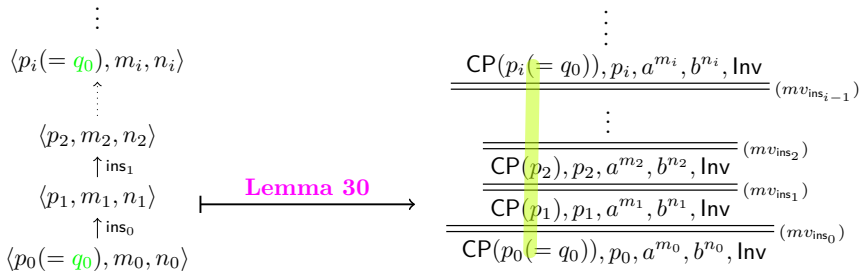


Figure 4 Simulation of an infinite run by a $\mu\mathsf{MALL}^\infty$ proof.

By assigning arbitrary polarities to atomic variables one can extend the notion to formulas in such a way that each formula is either positive or negative, depending on its top-level connective. A sequent is ***positive*** if it contains *only* positive or atomic formulas, otherwise it is ***negative***. A ***focused proof***, briefly, is one where bottom-up:

- only negative rules are applied on negative sequents; and,
- only positive rules are applied on positive sequents;
- any positive auxiliary formula of a positive rule must be principal for the next step;

Note that the focusing discipline described above ensure that, when reaching a positive sequent, bottom-up, positive rules are "hereditarily applied" on a particular positive formula, called the *focus*, until one reaches a negative sequent again. Importantly we have:

▶ **Theorem 29** ([6]). *If Γ has a cut-free $\mu\mathsf{MALL}^\infty$ proof, it also has one that is focused.*[4]

4.2 Provability implies run existence

In this subsection we prove the converse of Lemma 28 above:

▶ **Lemma 30.** *If the sequent $\nu x.\psi, q_0, \mathsf{Inv}$ is provable in $\mu\mathsf{MALL}^\infty$, then there exists a run of $\mathcal{M}$ from the configuration $\langle q_0, 0, 0\rangle$ such that q_0 is visited infinitely often.*

We shall henceforth assume that all $\mu\mathsf{MALL}^\infty$ proofs are cut-free, under Theorem 9, and focused, under Theorem 29. More specifically we assign atomic polarities as follows: a, b, z_a, z_b and q are negative for any state $q \in Q$. We first make a simple observation that will aid our proof.

▷ Claim 31. Inv is not principal in the lowest rule of any focused proof of $\psi, p, a^m, b^n, \mathsf{Inv}$.

Proof. The sequent $\psi, p, a^m, b^n, \mathsf{Inv}$ is positive so if Inv is active, then it is the focus. Without loss of generality, assume that the first rule is $(\oplus_1)$ with principal formula Inv. Then, the auxiliary formula is $(b^\perp)^* \otimes z_a$. Since the outermost connective is positive, we must immediately apply the $(\otimes)$ rule. One of the premisses is of the form $\Delta, z_a^\perp$ with $z_a^\perp$ as focus and we cannot apply any inference rule. Because Δ cannot be z_a, the identity rule is ruled out and no other rules are possible since z_a is an atom. ◁

We can now prove the main result of this subsection:

[4] The focusing result in [6] is for a logic without atoms but the proof technique can be straightforwardly extended to account for atoms.

Proof. The proof has two parts. We first show that one can carve out an infinite run $\mathcal{R}$ of $\mathcal{M}$ from $\langle q_0, 0, 0\rangle$ from a focussed proof of $\nu x.\psi, q_0, \mathsf{Inv}$. Then, we show that q_0 is visited infinitely often along $\mathcal{R}$.

Let P be a focussed proof of $\nu x.\psi, q_0, \mathsf{Inv}$. We claim that P can be factored as follows where $p_0 = q_0$, $m_0 = 0$, and $n_0 = 0$.

This factorisation yields the required infinite run $(\langle p_i, m_i, n_i\rangle)_{i\in\omega}$ where for all i,

$$\langle p_i, m_i, n_i\rangle \xrightarrow{\mathsf{ins}_i} \langle p_{i+1}, m_{i+1}, n_{i+1}\rangle.$$

Furthermore, if P is a proof, then there are infinitely many occurrences of $\nu x.\psi$ along this branch but, since $\mathsf{CP}(p_i) = \nu x.\psi$ only when $p_i = q_0$ we obtain that q_0 occurs infinitely often in the run. Therefore, we are left to prove that P can be factored as described.

We will give a proof-search argument to show that every pre-proof of $\mathsf{CP}(p), p, a^m, b^n, \mathsf{Inv}$ goes through $\mathsf{CP}(q), q, a^{m'}, b^{n'}, \mathsf{Inv}$ such that $\langle p, m, n\rangle \xrightarrow{\mathsf{ins}} \langle p', m', n'\rangle$ for some instruction ins. If $p = q_0$ and $\mathsf{CP}(p) = \nu x.\psi$ then the unique rule that can be applied is (ν) to obtain the sequent $\varphi, p, a^m, b^n, \mathsf{Inv}$ (otherwise $\mathsf{CP}(p)$ is anyway φ). From Claim 31, we get that Inv cannot be the focus. Therefore, φ is the focus and the next rules are necessarily (μ) and $\oplus$ respectively whence we have the sequent $[\mathsf{ins}], p, a^m, b^n, \mathsf{Inv}$ for some instruction ins. If ins is not an instruction that can be fired at p, proof-search immediately fails . If ins is an increment, it is trivial to obtain the result. If ins is a decrement of the form $p^\perp \otimes (((\mathsf{CP}(q) \parr q) \& z_a) \oplus (a^\perp \otimes (\mathsf{CP}(q') \parr q')))$, we need to make sure that the control goes to the appropriate state depending on whether r_1 is zero or not. We will show that an erroneous choice fails proof-search. We have two cases:

Case 1. Suppose we have $a^\perp \otimes (\mathsf{CP}(q') \parr q'), b^n, \mathsf{Inv}$. Here $a^\perp \otimes (\mathsf{CP}(q') \parr q')$ is the focus since in the earlier step $((\mathsf{CP}(q) \parr q) \& z_a) \oplus (a^\perp \otimes (\mathsf{CP}(q') \parr q'))$ was the focus. Therefore we have sequent of the form $\vdash \Delta, a^\perp$ where $a^\perp$ is the focus and Δ cannot be $\{a\}$.

Case 2. Suppose we have $(\mathsf{CP}(q) \parr q) \& z_a, a^m, b^n, \mathsf{Inv}$. This is a negative sequent, so the next rule is necessarily $(\&)$ and we have a premiss of the form $z_a, a^m, b^n, \mathsf{Inv}$ where Inv is the focus. It is easy to check that for choices $(\oplus_1)$ and $(\oplus_2)$, proof-search fails. ◀

Putting Lemmas 28 and 30 together we have:

▶ **Proposition 32** (Reduction). *$\mathcal{M}$ has an infinite run from q_0 visiting q_0 infinitely often if and only if there is a $\mu\mathsf{MALL}^\infty$ proof of $\nu x.\psi, q_0, \mathsf{Inv}$.*

By Theorem 25 we thus have:

▶ **Theorem 33.** *$\mu\mathsf{MALL}^\infty$ is Σ^1_1-hard.*

From here, we can conclude strictness of the inclusion from Theorem 14:

▶ **Corollary 34.** *$\mu\mathsf{MALL}^\infty$ and $\mu\mathsf{MALL}_{\omega,\infty}$ prove different sets of theorems.*

Proof. Clearly $\mu\mathsf{MALL}_{\omega,\infty} \in \Pi^1_1$: $\mu\mathsf{MALL}_{\omega,\infty}$ proves Γ just if:

"*every* set of sequents closed under $\mu\mathsf{MALL}_{\omega,\infty}$ rules contains Γ"

Note here that closure of a set X of sequents under $\mu\mathsf{MALL}_{\omega,\infty}$ is indeed arithmetical; in particular closure under the (ω)-rule is Π^0_2: "for *every* sequent $\Gamma, \nu x\varphi$ not in X there *exists* $n \in \omega$ such that $\Gamma, \varphi^n(\top)$ is not in X."

On the other hand, if $\mu\mathsf{MALL}^\infty = \mu\mathsf{MALL}_{\omega,\infty}$ then $\mu\mathsf{MALL}_{\omega,\infty}$ would be Σ^1_1-hard, by Theorem 33, contradicting its Π^1_1 membership as $\Sigma^1_1 \not\subseteq \Pi^1_1$. ◀

Finally Corollary 34 and Theorem 14 together imply Theorem 1, our main result.

5 A Π^1_2 upper bound on μMALL$^\infty$

Our Σ^1_1-hardness result, Theorem 33, places μMALL$^\infty$ definitively in the analytical hierarchy. Previously the best known lower bound was Π^0_1 from [13]. In terms of upper bounds, a naïve Σ^1_3 upper bound is readily obtained by the description of μMALL$^\infty$-provability:

"there *exists* a preproof s.t., *for all* infinite branches, there *exists* a progressing thread."

Note here that checking whether a given thread is progressing is indeed arithmetical: "there *exists* some $n \in \mathbb{N}$ and a formula $\nu x\varphi$ that is *infinitely often* principal, and such that *every* formula in the thread after position n has $\nu x\varphi$ as a subformula". In fact we can improve this upper bound considerably, comprising the main result of this section:

▶ **Theorem 35** ($\exists 0^\#$). μMALL$^\infty \in \Pi^1_2$.

Note that this result, strictly speaking, depends on the existence of $0^\#$ (as indicated), a principle independent of ZFC, but over which it is equivalent to lightface analytic determinacy [21]. To demonstrate this result we employ ideas from proof search, namely game theoretic formulations therein inspired by previous work [25, 18].

▶ **Definition 36** (Proof search game, for μMALL$^\infty$). *The* proof search game *for* μMALL$^\infty$ *is a two-player game played between Prover* (**P**)*, whose positions are inference steps of* μMALL$^\infty$*, and Denier* (**D**)*, whose positions are sequents of* μMALL$^\infty$*. A* play *of the game starts from a particular sequent: at each turn,* **P** *chooses an inference step with the current sequent as conclusion, and* **D** *chooses a premiss of that step; the process repeats from this sequent and the two players continue taking turns as long as possible.*

P wins *an infinite play of the game if the branch constructed has a progressing thread.*[5]

It is not hard to see that winning strategies for **P** correspond to non-wellfounded proofs:

▶ **Observation 37.** **P** *has a winning strategy from* Γ *iff there is a* μMALL$^\infty$ *proof of* Γ.

When the state space is finite, e.g. for the μ-calculus, the corresponding proof search game is finite-memory determined, yielding regular completeness of the proof system [33]. We do not have this property here, but the characterisation above nonetheless allows us to view **D** strategies as a form of "semantics" for μMALL$^\infty$ under determinacy:

▶ **Proposition 38** ($\exists 0^\#$). *The proof search game for* μMALL$^\infty$ *is determined.*

This is a consequence of (lightface) analytic determinacy, as the winning condition is indeed Σ^1_1: "there *exists* a progressing thread". From here we readily obtain our upper bound:

Proof of Theorem 35. There is a μMALL$^\infty$ proof of a sequent Γ if and only if **P** has a winning strategy from Γ by Observation 37, if and only if there is no winning strategy for **D** from Γ, by Proposition 38. The latter is clearly a Π^1_2 property:

"for *every* **D**-strategy there *exists* a play for which there *exists* a progressing thread" ◀

[5] In the case of deadlock, the player with no valid move loses.

6 Conclusion

In this work, we compared the expressivity of the infinitary systems $\mu\mathsf{MALL}^\infty$ and $\mu\mathsf{MALL}_{\omega,\infty}$ for linear logic with fixed points, and improved bounds on their complexity, cf. Figure 1. We conclude this paper with some remarks on potential future directions of research.

- It would be pertinent to extend our comparison to systems with *wider* branching, indexed by some ordinal α, say $\mu\mathsf{MALL}_{\alpha,\infty}$. Similar systems were considered in [16, 15]. Such systems become weaker (i.e. have fewer theorems) as α increases, as more cases must be proved to derive a ν formula. In this sense it would be particularly interesting if we could show that $\mu\mathsf{MALL}^\infty$ coincides with some $\mu\mathsf{MALL}_{\alpha,\infty}$, calibrating the strength of $\mu\mathsf{MALL}^\infty$ according to some ordinal measure. Let us point out that such an ordinal must be sufficiently large to evade a Π^1_1 upper bound, as for $\mu\mathsf{MALL}_{\omega,\infty}$, due to Σ^1_1-hardness of $\mu\mathsf{MALL}^\infty$; at the same time the systems $\mu\mathsf{MALL}_{\alpha,\infty}$ must reach a limit by $\alpha = \omega_1$, for cardinality reasons, giving a naïve upper bound.
- It would also be interesting to prove bona fide metalogical properties, such as cut-elimination and focusing, for $\mu\mathsf{MALL}_{\omega,\infty}$ (and friends), just like for $\mu\mathsf{MALL}^\infty$ in [6] and for several other infinitely branching systems in related areas [30, 24, 34]. Let us point out that the embedding of $\mu\mathsf{MALL}^\infty$ in $\mu\mathsf{MALL}_{\omega,\infty}$ of Section 3 does not introduce cuts, arguably evidence that $\mu\mathsf{MALL}_{\omega,\infty}$ might enjoy a well-behaved proof theory. We expect such a result to be easier to establish than the analogous results for $\mu\mathsf{MALL}^\infty$, thanks to the underlying wellfoundedness of $\mu\mathsf{MALL}_{\omega,\infty}$.
- What is the exact complexity of $\mu\mathsf{MALL}^\infty$? This question remains open after this work, but we have significantly narrowed the gap to the range between Σ^1_1 and Π^1_2. It would also be pertinent to investigate the complexity of the infinitary wellfounded system $\mu\mathsf{MALL}_{\omega,\infty}$ (and $\mu\mathsf{MALL}_{\omega,\omega}$ and friends). Let us point out also that the (weaker) Π^0_1 lower bound for $\mu\mathsf{MALL}^\infty$ from [13] applied already to the alternation-free fragment of $\mu\mathsf{MALL}^\infty$.[6] Our Σ^1_1 lower bound crucially uses a single alternation to mimic the Büchi condition on Minsky machines. It would be interesting to further investigate the effect of alternation on the complexity of systems we have investigated.
- Can the assumption of lightface analytic determinacy be dropped from Theorem 35? Attempting to do so has been beyond the scope of this work.

References

1 Bahareh Afshari, Sebastian Enqvist, and Graham E Leigh. Cyclic proofs for the first-order µ-calculus. *Logic Journal of the IGPL*, page jzac053, August 2022. doi:10.1093/jigpal/jzac053.

2 Bahareh Afshari and Graham E. Leigh. Cut-free completeness for modal mu-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005088.

3 R. Alur and T.A. Henzinger. A really temporal logic. In *30th Annual Symposium on Foundations of Computer Science*, pages 164–169, 1989. doi:10.1109/SFCS.1989.63473.

4 David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Logic*, 13(1), January 2012. doi:10.1145/2071368.2071370.

[6] "Alternation-free" means that no μ occurs under a ν and vice-versa.

5 David Baelde, Amina Doumane, Denis Kuperberg, and Alexis Saurin. Bouncing threads for circular and non-wellfounded proofs: Towards compositionality with circular proofs. In *LICS '22*, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533375.

6 David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.42.

7 David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In Nachum Dershowitz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 92–106, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

8 James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite descent. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 51–62. IEEE, 2007.

9 James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

10 Rudolf Carnap. *Logical Syntax of Language*. Kegan Paul, Trench and Truber, 1937.

11 Gianluca Curzi and Anupam Das. Non-uniform complexity via non-wellfounded proofs. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic, CSL 2023, February 13-16, 2023, Warsaw, Poland*, volume 252 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CSL.2023.16.

12 Anupam Das. On the logical complexity of cyclic arithmetic. *Log. Methods Comput. Sci.*, 16(1), 2020. doi:10.23638/LMCS-16(1:1)2020.

13 Anupam Das, Abhishek De, and Alexis Saurin. Decision problems for linear logic with least and greatest fixed points. In *FSCD*, volume 228 of *LIPIcs*, pages 20:1–20:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSCD.2022.20.

14 Anuj Dawar and Yuri Gurevich. Fixed Point Logics. *Bulletin of Symbolic Logic*, 8(1):65–88, 2002. doi:10.2178/bsl/1182353853.

15 Abhishek De. *Linear logic with the least and greatest fixed points: truth semantics, complexity, and a parallel syntax*. PhD thesis, Université Paris Cité, 2022. URL: https://www.irif.fr/_media/users/ade/main.pdf.

16 Abhishek De, Farzad Jafarrahmani, and Alexis Saurin. Phase semantics for linear logic with least and greatest fixed points. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 35:1–35:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.35.

17 Amina Doumane. *On the infinitary proof theory of logics with fixed points. (Théorie de la démonstration infinitaire pour les logiques à points fixes)*. PhD thesis, Paris Diderot University, France, 2017. URL: https://tel.archives-ouvertes.fr/tel-01676953.

18 Walter Felscher. Dialogues, strategies, and intuitionistic provability. *Annals of Pure and Applied Logic*, 28(3):217–254, 1985. doi:10.1016/0168-0072(85)90016-8.

19 Jérôme Fortier and Luigi Santocanale. Cuts for circular proofs: semantics and cut-elimination. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 248–262. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.248.

20 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987. doi:10.1016/0304-3975(87)90045-4.

21 Leo Harrington. Analytic determinacy and $0^{\#}$. *The Journal of Symbolic Logic*, 43(4):685–693, 1978. doi:10.2307/2273508.

22 Gerhard Jäger. Fixed points in peano arithmetic with ordinals. *Annals of Pure and Applied Logic*, 60(2):119–132, 1993. doi:10.1016/0168-0072(93)90039-G.

23 Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982. doi:10.1016/0304-3975(82)90125-6.

24 Dexter Kozen. A finite model theorem for the propositional μ-calculus. *Studia Logica*, 47(3):233–241, September 1988. doi:10.1007/BF00370554.

25 G Kreisel. P. lorenzen. ein dialogisches konstruktwitätskriterium. infinitistic methods, proceedings of the symposium on foundations of mathematics, warsaw, 2–9 september 1959, panstwowe wydawnictwo naukowe, warsaw, and pergamon press, oxford-london-new york-paris, 1961, pp. 193–200. *The Journal of Symbolic Logic*, 32(4):516–516, 1968.

26 Denis Kuperberg, Laureline Pinault, and Damien Pous. Cyclic proofs, system t, and the power of contraction. *Proc. ACM Program. Lang.*, 5(POPL):1–28, 2021. doi:10.1145/3434282.

27 Stepan Kuznetsov. Complexity of commutative infinitary action logic. In Manuel A. Martins and Igor Sedlár, editors, *Dynamic Logic. New Trends and Applications*, pages 155–169, Cham, 2020. Springer International Publishing.

28 Patrick Lincoln, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Ann. Pure Appl. Log.*, 56(1-3):239–311, 1992. doi:10.1016/0168-0072(92)90075-B.

29 R. Mansfield and G. Weitkamp. *Recursive Aspects of Descriptive Set Theory*. Oxford logic guides. Oxford University Press, 1985. URL: https://books.google.co.uk/books?id=jPzuAAAAMAAJ.

30 Grigori Mints. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics*, 10:548–596, 1978.

31 Michael Möllerfeld. *Generalized inductive definitions: the μ-calculus and Π^1_2-comprehension*. PhD thesis, Westfälische Universität Münster, 2002.

32 Yiannis N Moschovakis. *Elementary Induction on Abstract Structures (Studies in Logic and the Foundations of Mathematics)*. American Elsevier Pub. Co, 1974.

33 Damian Niwinski and Igor Walukiewicz. Games for the mu-calculus. *Theor. Comput. Sci.*, 163(1&2):99–116, 1996. doi:10.1016/0304-3975(95)00136-0.

34 Ewa Palka. An infinitary sequent system for the equational theory of *-continuous action lattices. *Fundam. Inf.*, 78(2):295–309, April 2007.

35 Gerald E. Sacks. *Higher Recursion Theory*. Perspectives in Logic. Cambridge University Press, 2017. doi:10.1017/9781316717301.

36 Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures*, volume 2303 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2002. doi:10.1007/3-540-45931-6_25.

37 Thomas Studer. On the proof theory of the modal μ-calculus. *Stud Logica*, 89(3):343–363, 2008. doi:10.1007/s11225-008-9133-6.

38 M. Y. Vardi. A temporal fixpoint calculus. In *POPL '88*, pages 250–259, New York, NY, USA, 1988. Association for Computing Machinery. doi:10.1145/73560.73582.

39 Igor Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. In *LICS 95, San Diego, California, USA, June 26-29, 1995*, pages 14–24, 1995.

Constraint LTL with Remote Access

Ashwin Bhaskar
Chennai Mathematical Institute, India

M. Praveen
Chennai Mathematical Institute, India
CNRS IRL ReLaX, Chennai, India

Abstract

Constraint Linear Temporal Logic (CLTL) is an extension of LTL that is interpreted on sequences of valuations of variables over an infinite domain. The atomic formulas are interpreted as constraints on the valuations. The atomic formulas can constrain valuations at the current position and positions that are a fixed distance apart (e.g., the previous position or the second previous position and so on). The satisfiability problem for CLTL is known to be PSPACE-complete. We generalize CLTL to let atomic formulas access positions that are unboundedly far away in the past. We annotate the sequence of valuations with letters from a finite alphabet and use regular expressions on the finite alphabet to control how atomic formulas access past positions. We prove that the satisfiability problem for this extension of the logic is decidable in cases where the domain is dense and open with respect to a linear order (e.g., rational numbers with the usual linear order). We prove that it is also decidable over integers with linear order and equality.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Modal and temporal logics; Theory of computation → Verification by model checking; Theory of computation → Automata over infinite objects

Keywords and phrases Constraint LTL, Regular Expressions, MSO formulas, Satisfiability, Büchi automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.41

Funding *M. Praveen*: This author is partially supported by the Infosys foundation.

1 Introduction

Propositional Linear Temporal Logic (LTL) and related automata-theoretic models have been extended in various ways to make it more expressive. Constraint LTL (abbreviated as CLTL in [12]), LTL with freeze operators [9], temporal logic of repeating values [7, 19], finite memory automata [16], data automata [3] are all examples of this. These extensions are concerned with using variables that range over infinite domains in place of Boolean propositions used in propositional LTL. Variables ranging over infinite domains are a natural choice for writing specifications for systems that deal with infinite domains. For example, CLTL has been used for specifications of cloud-based elastic systems [1], where the domain of natural numbers is used to reason about the number of resources that are being used by cloud-based systems. In CLTL, atomic formulas can refer to valuations in previous/next positions using operators like X^{-1}, X^{-2}, X, X^2 etc.

We generalize CLTL by introducing operators of the form $\textcircled{e}^{-1}$, where e is a regular expression over some finite alphabet. To illustrate the intended meaning of these operators, suppose $e = a^*b$. The operator $\textcircled{a^*b}^{-1}$ refers to a position in the past such that the sequence of letters between the current and that past position is in the language of the expression a^*b. The operator X^{-2} is same as $\textcircled{\Sigma \cdot \Sigma \cdot \Sigma}^{-1}$. We call this extension Constraint LTL with remote access (RCLTL). CLTL is a fragment of this, where the expressions e are constrained to be of the form Σ^n for some number n. Apart from being a natural extension of CLTL, RCLTL is inspired by the usage of automata to model smart contracts [18], where components interact

© Ashwin Bhaskar and M. Praveen;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 41; pp. 41:1–41:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

with each other by passing messages. For example, bidders in an online bidding forum may pass messages to propose a bidding value, or to request the current highest bid. Suppose we want to verify that any bid is always strictly greater than the previous bid. However, the previous bid may not have been proposed in the previous step: several past steps may have been requests for the current highest bid. If we represent bidding messages by the letter b and request messages by the letter r, we can access the position where the last bid was made using the operator $\textcircled{r^*b}^{-1}$. This operator can refer to past positions that are unboundedly far behind to access the last bid, which cannot be done in CLTL.

Contributions. The main contribution of this paper is to prove that RCLTL is decidable over the domain of integers with equality and linear order. At a high level, the techniques we use are similar to those used for proving decidability of satisfiability of CLTL [12], based on patterns that repeat in ultimately periodic words. However, the fact that RCLTL operators can access positions that are remotely in the past poses many challenges to be overcome. E.g., suppose a variable x has value 1 in the previous position and value 2 in the current position, so the constraint $X^{-1}x < x$ is satisfied. In CLTL, the values 1 and 2 are abstracted away and only the constraint $X^{-1}x < x$ is retained. This constraint spans two consecutive positions, referred to as a frame of length 2 in [12]. However in RCLTL, a constraint of the form $\textcircled{r^*b}^{-1}x < \textcircled{c}^{-1}x$ can span unboundedly many positions, so the concept of frames of fixed size cannot be used. We introduce memoirs to handle this. In turn, many properties of frames are not true for memoirs, so new techniques have to be introduced to handle them. Details of such differences with previous work will be highlighted at appropriate sections of this paper.

Related works. The satisfiability of the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{Z}_c)$ has been studied in [17] and has been proven to be EXPTIME-complete. The logic $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{Z}_c)$ subsumes CLTL $(\mathbb{Z}, <, =)$. In [11], the authors prove that the satisfiability of CTL* with constraints over $(\mathbb{Z}, <, =)$ is 2EXPTIME-complete. Again clearly, CTL* with constraints is an extension for CLTL $(\mathbb{Z}, <, =)$. In both these papers, the tree automata technique used is based on identifying repeating patterns over regular trees. The chapter "Functional Specification of Hardware via Temporal Logic" in [13] introduces an extension of LTL with regular expressions and a suffix implication operator.[1] This results in a temporal logic that extends LTL and is as expressive as ω-regular languages. Note however that, unlike in [13] where the authors extend the set of temporal operators, in our paper we extend the atomic formulas with regular expressions. Also note that the variables in our work range over infinite domains whereas [13] only deals with Boolean propositions. Atomic formulas are extended also in LTL with freeze operators [9] and temporal logic of repeating values [7, 19]. Like our extension, the atomic formulas there can also compare current values with values in other positions that are unboundedly far away. However, in our extension the other position is chosen by a pattern of letters from a finite alphabet, whereas the atomic formulas in LTL with freeze operators and temporal logic of repeating values can choose other positions based on the values from the infinite domain. For example, in LTL with freeze operators, an atomic formula can check that there is some position in the past where the value is equal to the value at the current position. The negation of this formula checks the current value with all the previous values. In the logic we propose here, only two positions can be compared. This feature of LTL with freeze operators is quite powerful and satisfiability is undecidable in general, while the logic we propose here is decidable.

[1] We thank an anonymous reviewer for pointing this work to us.

Temporal Stream Logic (TSL) [15, 14] extends LTL with updates and predicates over arbitrary function terms. TSL is designed to track how variable values evolve over time. The satisfiability problem for TSL is not decidable but approaches to be used in practice have been proposed. Constraint LTL over clocks [2] is a variant of CLTL and many other time-domain extensions of LTL (such as Metric Interval Temporal Logic and Quantified Temporal Logic) can be translated to CLTL over clocks. The satisfiability problem for this logic has been shown to be decidable, by reducing it to a decidable Satisfiability Modulo Theory problem. Presburger LTL is an extension of LTL where atomic formulas are quantifier-free Presburger formulas. The satisfiability problem for this logic is undecidable in general. Many decidable fragments have been proposed [8, 6, 4]. The survey [10] mentions many other logics that are extended at the atomic level to deal with concrete domains.

2 Preliminaries

Let $\mathbb{Z}$ be the set of integers and $\mathbb{N}$ be the set of non-negative integers. For integers n_1, n_2, we denote by $[n_1, n_2]$ the set $\{n \in \mathbb{Z} \mid n_1 \leq n \leq n_2\}$. We recall the definitions of constraint systems and Constraint Linear Temporal Logic (CLTL) from [12]. A constraint system $\mathcal{D}$ is of the form $(D, R_1, \ldots, R_n, \mathcal{I})$, where D is a non-empty set called the domain. Each R_i is a predicate symbol of arity a_i, with $\mathcal{I}(R_i) \subseteq D^{a_i}$ being its interpretation.

We introduce an extension of Constraint LTL (CLTL) to let atomic formulas access positions in the past that are unboundedly far away. Let V be a finite set of variables, Σ be a finite alphabet and let E be a finite set of regular expressions over Σ. A term t over V and E is of the form $ⓔ^{-1}v$, where $v \in V$, $e \in E$[2]. Let T_E^V denote the set of all terms over V and E. A constraint c is of the form $R(t_1, \ldots, t_n)$, where R is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms. The syntax of Constraint LTL with Remote access (RCLTL) is given by the following grammar, where c is a constraint as defined above.

$$\phi ::= c \mid \neg\phi \mid \phi \vee \phi \mid X\phi \mid \phi U \phi$$

Let F_D denote the set of all mappings of the form $f\colon V \to D$. The semantics of RCLTL is defined with respect to infinite sequences over $\Sigma \times F_D$ (also called concrete models in the following). Given a concrete model σ, a regular expression $e \in E$ and positions $j, i \in \mathbb{N}$, we write $\sigma[j,i] \in \mathcal{L}(e)^R$ to denote that the infix of σ between the positions j and i (both j and i inclusive), projected to Σ is in the reverse of the language of e. We say that j is the minimal match for e at i if $\sigma[j,i] \in \mathcal{L}(e)^R$ and for every $j < j' \leq i$, $\sigma[j',i] \notin \mathcal{L}(e)^R$ (the word "minimal" refers to minimality with respect to the length of infixes ending at position i). In other words, $\sigma[j,i]$ belongs to $\mathcal{L}(e)^R$, but none of its suffixes belong to $\mathcal{L}(e)^R$. A regular expression $e \in E$ is said to be relevant at position i of a concrete model if there exists a position $j \geq 0$ that is the minimal match for e at i. A term $t = ⓔ^{-1}v$ is said to be relevant at a position i if e is relevant at i. Given a position i and a term $ⓔ^{-1}v$ relevant at i, we write $d(i,e,v)$ (resp. $\mathrm{mmp}(i,e)$) to denote the value $f_j(v) \in D$ (resp. j) such that j is the minimal match for e at i. Intuitively, if we start at position i, go backwards and j is the first position such that the infix $\sigma[j,i]$ belongs to $\mathcal{L}(e)^R$, then $d(i,e,v)$ is the value assigned to v at that position j and $\mathrm{mmp}(i,e) = j$, the minimal matching position.

[2] This choice of notation is inspired by a convention of LTL syntax, where $O\phi$ is intended to mean next ϕ. We write O^{-1} to mean past and insert e inside the O to indicate that we want the path between the current and past positions to match e, similar to the syntax used in dynamic logics.

Given $v_1, \ldots, v_n \in V$ and $e_1, \ldots, e_n \in E$, the ith position of a concrete model σ satisfies the constraint $R(\textcircled{e_1}^{-1}v_1, \ldots, \textcircled{e_n}^{-1}v_n)$ (written as $\sigma, i \models R(\textcircled{e_1}^{-1}v_1, \ldots, \textcircled{e_n}^{-1}v_n)$) if $e_1, \ldots, e_n$ are all relevant at i and $(d(i, e_1, v_1), \ldots, d(i, e_n, v_n)) \in \mathcal{I}(R)$. Intuitively, the terms $\textcircled{e_1}^{-1}v_1, \ldots, \textcircled{e_n}^{-1}v_n$ refer to variables at previous positions and $R(\textcircled{e_1}^{-1}v_1, \ldots, \textcircled{e_n}^{-1}v_n)$ imposes a constraint on the values of those variables at those positions. Note that $\sigma, i \models \textcircled{e}^{-1}v = \textcircled{e}^{-1}v$ iff e is relevant at i. E.g., the expression b^+a is relevant at position i if "b" is the letter at i, $j < i$ is the last position before i that has "a" and all positions between j and i carry "b"; j is the minimal matching position.

The semantics is extended to rest of the RCLTL syntax similar to the usual propositional LTL. We use the standard abbreviations $F\phi$ (resp. $G\phi$) to mean that ϕ is true at some position (resp. all positions) in the future. The formula $XG(\textcircled{b^*a}^{-1}v < \textcircled{a^*b}^{-1}v)$ will be true if in all positions other than the first, the value of v at the last position with an "a" is less than the value at the last position with "b". In the rest of this paper we will only consider constraint systems of the form $(D, <, =)$ where D is an infinite set, $=$ is the equality relation and $<$ is a linear order. Some proofs and technical details in the subsequent sections are moved to the appendix due to space constraints.

3 Symbolic Models

CLTL (introduced in [12]) can be considered as a restriction of RCLTL where the expressions in E are restricted to be of the form Σ^n for some $n \in \mathbb{N}$. The maximum such n used in a given CLTL formula is called the X-length of the formula. The models of CLTL are infinite sequences over an infinite alphabet. The notion of frames was introduced in [12] in order to abstract the infinite alphabet to a finite one. A frame for a given CLTL formula, as introduced in [12], is the set of all constraints satisfied by valuations along some infix of some concrete model, the length of the infix being bound by the X-length of the formula. The spans of these frames are equal to the X-length of the CLTL formula. In a fundamental deviation from [12], we introduce here the notion of a *memoir*, which can span positions that are unboundedly far away from one another.

▶ **Definition 1** (Memoirs and symbolic models). *Let ϕ be an RCLTL formula with set of variables V and expressions E. A* memoir *is a total pre-order*[3] *$\leq$ on a subset of terms over V and E. A* symbolic model *ρ is an infinite sequence in $(\Sigma \times M)^\omega$, where M is the set of all memoirs.*

Given a concrete model σ, we associate with it a symbolic model $\mu(\sigma)$ as follows: $\mu(\sigma)$ is an infinite sequence such that for every $i \geq 0$, $e_1, e_2 \in E$ and $v_1, v_2 \in V$, the ith memoir has the relation $\textcircled{e_1}^{-1}v_1 \leq_i \textcircled{e_2}^{-1}v_2$ iff e_1 and e_2 are relevant at position i and $d(i, e_1, v_1) \leq d(i, e_2, v_2)$. Intuitively, the memoir at position i of $\mu(\sigma)$ stores in its memory the relations between all variables in those past positions which can be accessed from position i through one of the given regular expressions.

For a symbolic model ρ, let $\leq_i$ be the memoir at position i. We denote by $<_i$ and $\equiv_i$ the strict order and equivalence relation induced by $\leq_i$: $t_1 <_i t_2$ iff $t_1 \leq_i t_2$ and $t_2 \not\leq_i t_1$, and $t_1 \equiv_i t_2$ iff $t_1 \leq_i t_2$ and $t_2 \leq_i t_1$. We define symbolic semantics $\models_s$, to interpret RCLTL formulas on symbolic models. Minimal matches in symbolic models are just like those in concrete models: we say that j is the minimal match for expression e at position i of a symbolic model ρ if $\rho[j, i] \in \mathcal{L}(e)^R$ and for every $j < j' \leq i$, $\rho[j', i] \notin \mathcal{L}(e)^R$. The definitions

[3] a reflexive and transitive relation such that for all a, b, either $a \leq b$ or $b \leq a$.

of the relevance of regular expressions and terms are the same for both symbolic and concrete models. The ith position of ρ symbolically satisfies the constraint $t_1 < t_2$ (where t_1, t_2 are terms), if t_1, t_2 are relevant at i and $t_1 <_i t_2$ holds. Formally, we write it as $\rho, i \models_s t_1 < t_2$. The case of the equality constraint is similar. The symbolic satisfaction relation $\models_s$ is extended to all RCLTL formulas by induction on the structure of the formula as done for propositional LTL. Following is the RCLTL counterpart of a similar result [12, Lemma 3.1] for CLTL. The proof is by a routine induction on the structure of formulas.

▶ **Lemma 2.** *Let ϕ be an RCLTL($\mathcal{D}$) formula. Let σ be a concrete model and $\rho = \mu(\sigma)$. Then $\sigma, 0 \models \phi$ iff $\rho, 0 \models_s \phi$.*

For every concrete model, there is an associated symbolic model. However, there may be symbolic models that are not associated with any concrete models – symbolic models are intended to be abstractions of concrete models, but some combination of abstract symbols may not make sense. For example, the symbolic model ρ may have memoirs that are *irrelevant*: the memoir at some position i may have the relation $\textcircled{e_1}^{-1} v_1 \leq_i \textcircled{e_2}^{-1} v_2$, but there may not be any position that is a match for e_1 at i. We say that a symbolic model ρ admits a concrete model if there exists a concrete model σ such that $\rho = \mu(\sigma)$.

▶ **Definition 3.** ***Satisfiability Problem for RCLTL****: Given an RCLTL($\mathcal{D}$) formula ϕ, the satisfiability problem is to check if there exists a concrete model that satisfies ϕ*

From Lemma 2, we have the following corollary.

▶ **Corollary 4** ([12, Corollary 4.1])**.** *An RCLTL($\mathcal{D}$) formula ϕ is satisfiable iff there exists a symbolic model ρ such that $\rho, 0 \models_s \phi$ and ρ admits a concrete model.*

To check whether $\rho, i \models_s t_1 < t_2$ we only need to check $\rho(i)$, the ith memoir in ρ, unlike the RCLTL semantics, where we may need to check other positions also. In this sense, the symbolic semantics lets us treat RCLTL formulas as if they are formulas in propositional LTL and employ techniques that have been developed for propositional LTL. To get decidability of RCLTL satisfiability, we additionally need to check if symbolic models admit concrete ones. For this, as mentioned before, we first need to check that memoirs in symbolic models are relevant.

Another reason why symbolic models may not admit concrete ones is that the symbolic model may be inconsistent: one memoir may say that the value of variable x at position i should be less than the value of y at position j, but another memoir may say the opposite (i.e., value of x at i should be greater than the value of y at j). To explain how we tackle this, we use a graph associated with symbolic models, as done in [12]. For a symbolic model ρ, we define a corresponding $\{<, =\}$-labelled, directed graph G_ρ over the set of vertices $V \times \mathbb{N}$. There is a $\sim$-labelled edge (for $\sim \in \{<, =\}$) from (x, i) to (y, j) (written as $(x, i) \xrightarrow{\sim} (y, j)$) if and only if there exists $k \geq i, j$ and there exist $e_1, e_2 \in E$ such that i is the minimal match for e_1 at k, j is the minimal match for e_2 at k and the constraint $\textcircled{e_1}^{-1} x \sim_k \textcircled{e_2}^{-1} y$ holds. Intuitively, the total pre-order at position k imposes the constraint $\sim$ between the variable x at position i and the variable y at position j. It follows that if there is an $=$-labelled edge from (x, i) to (y, j) then there must also be an $=$-labelled edge from (y, j) to (x, i). We define a strict edge to be an edge labelled with $<$ and a directed path is said to be strict if it contains at least one strict edge.

We say that G_ρ has a strict cycle if it has a directed cycle containing at least one edge labelled with $<$. Strict cycles indicate the presence of inconsistencies in symbolic models, as explained before. If G_ρ is the graph associated with a symbolic model ρ, then we need

to check if between every pair of vertices in G_ρ there is at most one directed edge labelled with $<$, and that there are no strict cycles. Notice here that given a concrete model σ, $G_{\mu(\sigma)}$ does not have any strict cycle. Depending on the letters from Σ present along the positions of a symbolic model, the minimal match for some expression e at some position i can be unboundedly far behind i. So the presence/absence of an edge between a pair of vertices in G_ρ can potentially depend on memoirs that are unboundedly far in the future. This is a deviation from [12], whose frames can only span positions bounded by the X-length of the given formula (so consistency checks only need to be performed at bounded lengths).

Given a symbolic model ρ, it is easy to see that ρ admits a concrete model over a domain D iff there exists a labelling $l\colon V \times \mathbb{N} \to D$ of the vertices of G_ρ such that the labelling respects the edges in G_ρ. In other words, for all $x, y \in V, i, j \in \mathbb{N}$, $l((x,i)) = l((y,j))$ if there exists a directed path from (x,i) to (y,j) consisting of only $=$-labelled edges and $l((x,i)) < l((y,j))$ if there exists a strict directed path from (x,i) to (y,j). We call such a labelling of vertices in G_ρ, an edge-respecting labelling.

We say a domain D is dense (with respect to the ordering $<$) if for each $d, d' \in D$ with $d < d'$, there exists $d'' \in D$, such that $d < d'' < d'$. We say D is open if for each $d \in D$, there exist $d', d'' \in D$ such that $d' < d < d''$. $(\mathbb{R}, <, =)$ and $(\mathbb{Q}, <, =)$ are examples of constraint systems where the domains are dense and open. We say a symbolic model is consistent if all its memoirs are relevant and G_ρ does not have strict cycles. Using the proof ideas from Lemma 5.3 and Lemma 6.1 of [12], we get the following lemma:

▶ **Lemma 5** ([12, Lemma 5.3, Lemma 6.1]). *Let $\mathcal{D}$ be a constraint system of the form $(D, <, =)$ where D is infinite, $<$ is a linear order and D is dense and open with respect to $<$. Then all consistent symbolic models admit concrete models over D.*

In the next section we will give an MSO characterization of symbolic models with all memoirs relevant and whose associated graphs G_ρ have no strict cycles. This will help us in deciding the satisfiability problem for RCLTL($\mathcal{D}$) over dense and open domains.

4 A general decidability result

In this section, we give a decision procedure for the satisfiability problem for RCLTL over domains that are dense and open. Corollary 4 suggests a way to address the satisfiability problem for RCLTL. Given an RCLTL formula ϕ, we know that we can construct a Büchi automaton that recognizes the set of all symbolic models over $\Sigma \times M$ which symbolically satisfy ϕ [21]. If we could build two Büchi automata, one that recognizes the set of symbolic models with all memoirs relevant and with no strict cycle in the associated graph, and one that recognizes symbolic models that admit a concrete model, then we can intersect these three automata and check the resulting automaton for non-emptiness to decide the satisfiability for ϕ.

There are constraint systems for which the set of symbolic models that admit concrete models is not ω-regular. However, as we saw in Lemma 5, for constraint systems whose domains are dense and open, every consistent symbolic model admits a concrete model, hence constructing an automaton that accepts the set of consistent symbolic models suffices.

We now give a characterization of consistent symbolic models in the following lemma:

▶ **Lemma 6.** *Let $\mathcal{L}_{symb}$ denote the set of all consistent symbolic models ρ over $(\Sigma \times M)^\omega$. Then, $\mathcal{L}_{symb}$ is ω-regular.*

Proof. We prove that $\mathcal{L}_{symb}$ is ω-regular by constructing an MSO formula over the alphabet $\Sigma \times M$ defining $\mathcal{L}_{symb}$. We shall define two MSO formulas: ψ_{rel} whose language is the set of all symbolic models with all memoirs relevant and $\psi_{strcycle}$ whose language is the set of all symbolic models ρ whose associated graphs G_ρ have no strict cycles.

We first define the unary predicate symbols in the MSO vocabulary and their semantics. Let the set of predicate symbols be $P = \{P_a \mid a \in \Sigma\} \cup \{P_{\textcircled{e_1}^{-1}v_j \sim \textcircled{e_2}^{-1}v_k} \mid e_1, e_2 \in E, v_j, v_k \in V, \sim \in \{<, =\}\}$. Given a symbolic model ρ and a position i, $\rho, i \models i \in P_a$ iff $\rho(i)$ is of the form $(a, \leq_i)$ where $\leq_i$ is any memoir in M. We use $P_{\textcircled{e_1}^{-1}v_j \leq \textcircled{e_2}^{-1}v_k}$ as a shorthand for $P_{\textcircled{e_1}^{-1}v_j < \textcircled{e_2}^{-1}v_k} \vee P_{\textcircled{e_1}^{-1}v_j = \textcircled{e_2}^{-1}v_k}$. We also say that an edge of G_ρ is labelled with $\leq$ if it is labelled either with $<$ or $=$. We have $\rho, i \models i \in P_{\textcircled{e_1}^{-1}v_j \sim \textcircled{e_2}^{-1}v_k}$ iff the memoir at $\rho(i)$ induces the relation $\textcircled{e_1}^{-1}v_j \sim \textcircled{e_2}^{-1}v_k$.

We now define a few MSO formulas that will aid us in constructing the formula ψ_{rel}.

Let $\text{matches}_e(pos, match)$ be an MSO formula which evaluates to true iff the position $match$ is a match for e at pos, in other words, if $\rho[pos, match]^R \in \mathcal{L}(e)$. Such a formula can be easily constructed using standard techniques for translating automata to MSO formulas.

For a regular expression $e \in E$ and two positions pos and $match$, the following formula $\text{minmatch}_e(pos, match)$ is true iff the position $match$ is a minimal match for e at pos.

$$\text{minmatch}_e(pos, match) \equiv \text{matches}_e(pos, match) \wedge \\ \forall x((match < x \leq pos) \implies \neg(\text{matches}_e(pos, x)))$$

Given a regular expression $e \in E$, and given a position $pos \in \mathbb{N}$, we have the following formula $\text{rel}_e(pos)$ that is true iff e is relevant at pos.

$$\text{rel}_e(pos) \equiv \exists\, match(\text{minmatch}_e(pos, match))$$

We now define formula ψ_{rel}.

$$\psi_{rel} = \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4 \wedge \psi_5$$

where ψ_1 denotes that every position contains exactly one letter.

$$\psi_1 \equiv \forall x(\bigvee_{a \in \Sigma} (x \in P_a) \wedge \bigwedge_{a_i, a_j \in \Sigma, i \neq j} \neg(x \in P_{a_i} \wedge x \in P_{a_j}))$$

The formula ψ_2 says that given a pair of terms t_1, t_2 at a position x, if the predicate $P_{t_1 = t_2}$ holds at x, then the predicate $P_{t_2 = t_1}$ must also hold at x.

$$\psi_2 \equiv \forall x \bigwedge_{e_1, e_2 \in E} \bigwedge_{v_i, v_j \in V} ((x \in P_{\textcircled{e_1}^{-1}v_j = \textcircled{e_2}^{-1}v_k}) \Leftrightarrow (x \in P_{\textcircled{e_2}^{-1}v_k = \textcircled{e_1}^{-1}v_j}))$$

The formula ψ_3 says that given a pair of terms t_1, t_2 at a position x, at most one predicate among $P_{t_1 < t_2}$, $P_{t_1 = t_2}$ and $P_{t_2 < t_1}$ comparing this pair of terms holds at x.

$$\psi_3 \equiv \forall x \bigwedge_{e_1, e_2 \in E} \bigwedge_{v_i, v_j \in V} ((\neg(x \in P_{\textcircled{e_1}^{-1}v_j < \textcircled{e_2}^{-1}v_k} \wedge x \in P_{\textcircled{e_1}^{-1}v_j = \textcircled{e_2}^{-1}v_k})) \\ \wedge (\neg(x \in P_{\textcircled{e_2}^{-1}v_k < \textcircled{e_1}^{-1}v_j} \wedge x \in P_{\textcircled{e_1}^{-1}v_j < \textcircled{e_2}^{-1}v_k})))$$

The formula ψ_4 says that if e_1 and e_2 are relevant expressions at a position x, then the memoir at x must compare every term $\textcircled{e_1}^{-1}v_i$ with every term $\textcircled{e_2}^{-1}v_j$.

$$\psi_4 \equiv \forall x(\bigwedge_{e_1, e_2 \in E} (\ (\text{rel}_{e_1}(x) \wedge \text{rel}_{e_2}(x)) \implies \\ \bigwedge_{v_i, v_j \in V} ((x \in P_{\textcircled{e_1}^{-1}v_i \leq \textcircled{e_2}^{-1}v_j}) \vee (x \in P_{\textcircled{e_2}^{-1}v_j \leq \textcircled{e_1}^{-1}v_i}))))$$

The formula ψ_5 says that at every position x, the memoir at x compares only terms that are relevant at x.

$$\psi_5 \equiv \forall x(\bigwedge_{e_1,e_2 \in E} (\bigvee_{v_i,v_j \in V} ((x \in P_{\textcircled{e_1}^{-1}v_i \leq \textcircled{e_2}^{-1}v_j}) \vee (x \in P_{\textcircled{e_2}^{-1}v_j \leq \textcircled{e_1}^{-1}v_i})) \implies (\mathrm{rel}_{e_1}(x) \wedge \mathrm{rel}_{e_2}(x))))$$

It is now easy to verify that the formula ψ_{rel} is satisfied by exactly those symbolic models, all of whose memoirs are relevant.

We now define a few MSO formulas that will aid us in constructing the formula $\psi_{strcycle}$.

Given $\sim \in \{\leq, =\}$, the formula $\sim_{i,j}(x,y)$ is true iff there is a $\sim$-labelled edge from (v_i, x) to (v_j, y) in G_ρ. It is defined as:

$$\sim_{i,j}(x,y) \equiv \exists z((z \geq x) \wedge (z \geq y) \wedge \bigvee_{e_1,e_2 \in E} z \in P_{\textcircled{e_1}^{-1}v_i \sim \textcircled{e_2}^{-1}v_j} \wedge \mathrm{minmatch}_{e_1}(z,x) \wedge \mathrm{minmatch}_{e_2}(z,y))$$

Let $l = |V|$. Given $\sim \in \{\leq, =\}$, we define the following formulas:

$$\mathrm{closed}_k^{\sim}(y, Y_1, \ldots, Y_l) \equiv \bigwedge_{p,p' \in \{1,\ldots,l\}} \forall z \forall z'((z \in Y_p \wedge \sim_{p,p'} (z,z')) \implies z' \in Y_{p'}) \wedge \forall z(\bigwedge_{k' \in \{1,\ldots,l\}} (\sim_{k,k'} (y,z) \implies z \in Y_{k'}))$$

$$\mathrm{path}_{i,j}^{\sim}(x,y) \equiv \forall Y_1 \ldots \forall Y_l(\mathrm{closed}_i^{\sim}(x, Y_1, \ldots, Y_l) \implies y \in Y_j)$$

The intention of the formula $\mathrm{closed}_k^{\sim}(y, Y_1, \ldots, Y_l)$ is the following: if there is a path starting from variable v_k at position y to some variable v_j at some position z, then z must belong to Y_j. This can be proved by an induction on the length of the path in addition to Knaster-Tarski theorem over the lattice of l-tuples of subsets of $\mathbb{N}$.

We claim that the formula $\mathrm{path}_{i,j}^{\sim}(x,y)$ holds iff there is a directed path of non-zero length from (v_i, x) to (v_j, y) in G_ρ, all of whose edges are labelled with $\sim$. The details of the proof can be found in section B of the appendix.

We define a new formula $\mathrm{path}_{i,j}^{<}(x,y)$ which holds iff there is a strict directed path from (v_i, x) to (v_j, y).

$$\mathrm{path}_{i,j}^{<}(x,y) \equiv \mathrm{path}_{i,j}^{\leq}(x,y) \wedge \neg \mathrm{path}_{i,j}^{=}(x,y)$$

We now define the MSO formula $\psi_{strcycle}$.

$$\psi_{strcycle} \equiv \neg(\mathrm{path}_{i,i}^{<}(x,x))$$

Thus, we have a formula $\psi_{symb} \equiv \psi_{rel} \wedge \psi_{strcycle}$ whose language is the set of all consistent symbolic models. Hence, $\mathcal{L}_{symb} = \mathcal{L}(\psi_{symb})$. Now, by Büchi's theorem [5], we know that we can construct a Büchi automaton $\mathcal{A}_{symb}$ recognizing $\mathcal{L}_{symb}$, or in other words, $\mathcal{L}_{symb}$ is ω-regular. ◀

▶ **Lemma 7.** *Let $\mathcal{D} = (D, <, =)$ be a constraint system such that D is dense and open. Let ϕ be an RCLTL($\mathcal{D}$) formula. Then, the language $\mathcal{L} = \{\mu(\sigma) \mid \sigma \models \phi\}$ is ω-regular.*

Proof. Lemma 5 and Corollary 4 imply that $\mathcal{L} = \mathcal{L}_{symb} \cap \mathcal{L}^{\phi}_{sat}$ where $\mathcal{L}^{\phi}_{sat}$ is equal to the set of symbolic models that symbolically satisfy ϕ. A straightforward adaptation of standard automata-theoretic construction for propositional LTL [21] proves that $\mathcal{L}^{\phi}_{sat}$ is ω-regular. Also, we know that $\mathcal{L}_{symb}$ is ω-regular, as shown before. Since ω-regular languages are closed under intersection, $\mathcal{L}$ is also ω-regular. ◀

▶ **Theorem 8.** *The satisfiability problem for RCLTL($\mathcal{D}$) where $\mathcal{D} = (D, <, =)$ and D is a dense and open domain, is decidable*

Proof. Solving the satisfiability problem for RCLTL($\mathcal{D}$) amounts to checking if, given an RCLTL($\mathcal{D}$) formula ϕ, $\mathcal{L} = \{\mu(\sigma) \mid \sigma \models \phi\} \neq \emptyset$. By Lemma 7, we know $\mathcal{L}$ is ω-regular. Hence, it is possible to construct a Büchi automaton accepting $\mathcal{L}$ and check for its non-emptiness. ◀

5 Satisfiability over $(\mathbb{Z}, <, =)$

In this section we consider the constraint system $\mathcal{Z} = (\mathbb{Z}, <, =)$. Since $\mathbb{Z}$ is not dense, Lemma 5 does not apply here. We solve the satisfiability problem for RCLTL over $(\mathbb{Z}, <, =)$ by using an automata-theoretic technique, similar to the one used in [12] to solve the satisfiability problem for CLTL over $(\mathbb{Z}, <, =)$. The idea is to define a Büchi automaton which accepts a superset of the set of all symbolic models which admit a concrete model with the property that all ultimately periodic words recognized by this automaton admit concrete models.

We begin with a characterization of symbolic models that admit a concrete model over $\mathbb{Z}$. Let ρ be a consistent symbolic model. For a directed path p in G_ρ, let $slen(p)$ denote the strict length of p, i.e. the number of strict edges in p if this number is finite, and ω otherwise. For any two vertices $u, v \in G_\rho$, define $slen(u, v)$ to be the supremum of $slen(p)$ over directed paths p from u to v, if it exists, and ω otherwise. If there is no direct path from u to v, $slen(u, v)$ is 0.

▶ **Lemma 9** ([12, Lemma 6.1]). *A consistent symbolic model ρ admits a concrete model over $\mathbb{Z}$ iff for all $u, v \in G_\rho$, $slen(u, v) < \omega$.*

Next we would like to design finite state automata to detect whether there are vertices u, v with $slen(u, v) = \omega$. Finite state automata cannot remember paths of unbounded lengths, so we look for bounded paths which can be repeated arbitrarily many times to get unbounded $slen$. The central idea in [12] is to show that such repeatable bounded paths are guaranteed to exist in ultimately periodic sequences. We borrow that idea here; but to implement that idea here requires much work. We now introduce some notation, conditions and results to formalize this.

Given a symbolic model ρ, and given a vertex (x, i) of G_ρ, we define the "level" of vertex (x, i) to be i and the "var" of vertex (x, i) to be x. A forward $u - v$ (resp. backward $u - v$) witness in G_ρ is a sequence $\nu \colon \mathbb{N} \to V \times \mathbb{N}$ such that $\nu(0) = u$ (resp. v), for all $i \geq 0$ there is a strict path from $\nu(i)$ to $\nu(i+1)$ (resp. from $\nu(i+1)$ to $\nu(i)$), $\text{level}(\nu(i+1)) > \text{level}(\nu(i))$ and for infinitely many i, there is a directed path from $\nu(i)$ to v (resp. from u to $\nu(i)$). We now give a condition $C_{\mathbb{Z}}$ (similar to condition $C_{\mathbb{Z}}$ in [12]) on a symbolic model ρ.

($C_{\mathbb{Z}}$): There do not exist vertices $u, v \in G_\rho$ with either a forward or backward $u - v$ witness.

If ρ does not satisfy $C_{\mathbb{Z}}$, there exist vertices u and v with either a forward or a backward $u - v$ witness implying that $slen(u, v) = \omega$. Figure 1 illustrates a forward $u - v$ witness. Hence, lemma 9 implies that ρ does not admit a concrete model over $\mathbb{Z}$. We shall now see

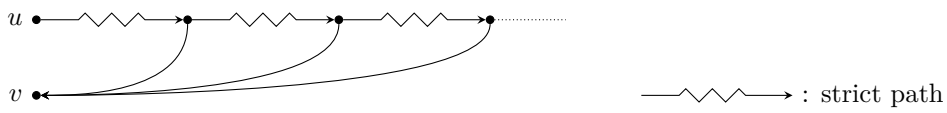


Figure 1 A forward $u - v$ witness.

that if ρ is an ultimately periodic word, then $C_{\mathbb{Z}}$ is both necessary and sufficient. We say a symbilic model is ultimately periodic if it is of the form $\tau \cdot \delta^\omega$ for some finite words τ and δ over $\Sigma \times M$.

We use the following terminologies with respect to an ultimately periodic consistent symbolic model $\rho = \tau \cdot \delta^\omega$. A vertex in G_ρ is said to be a τ-vertex if its level is at most $|\tau|$. The suffixes $\rho[i, \infty)$ and $\rho[i + m|\delta|, \infty)$ are equal for every $m \in \mathbb{N}$ and $i > |\tau|$. Accordingly, we say the pair of vertices (x, i) and $(x, i + m|\delta|)$ are isomorphic for all $i > |\tau|$ and all $m \in \mathbb{N}$ and denote it by $(x, i) \cong (x, i + m|\delta|)$. With slight abuse of notation, we also say that the positions i and $i + m|\delta|$ are isomorphic. The set of isomorphism classes $\cong_0, \cong_1, \cdots, \cong_{|\delta|-1}$ is a partition of $[|\tau|+1, \infty)$ such that for all $m \in \mathbb{N}$ and for all $j \in [0, |\delta|-1]$, any position of the form $|\tau|+m|\delta|+j$ is in the isomorphism class $\cong_j$. A non-τ-vertex v in G_ρ is said to be in the isomorphism class $\cong_j$ if $\text{level}(v) \in \cong_j$. An edge from or to a τ-vertex is said to be a long jump. An edge between non-τ-vertices is said to be a hop. It is a strict hop if it is labelled by $<$. A path is said to be a hopping path if it does not contain any long jumps. A forward hopping map of n traversals is a sequence $\mu_{fh} : [0, n] \to V \times \mathbb{N}$ such that for every $i \in [0, n-1]$, there is a hopping path from $\mu_{fh}(i)$ to $\mu_{fh}(i+1)$ and $\text{level}(\mu_{fh}(i+1)) > \text{level}(\mu_{fh}(i))$. The rank of a forward hopping map μ_{fh} is the number of indices i such that there is a strict path from $\mu_{fh}(i)$ to $\mu_{fh}(i+1)$. The existence of one strict path from $\mu_{fh}(i)$ to $\mu_{fh}(i+1)$ ensures that all paths from $\mu_{fh}(i)$ to $\mu_{fh}(i+1)$ are strict – if there was a strict path and a non-strict path from $\mu_{fh}(i)$ to $\mu_{fh}(i+1)$, combining them would result in a strict cycle from $\mu_{fh}(i)$ to itself, contradicting the consistency of G_ρ. From a forward hopping map μ_{fh} of n traversals, we can obtain a path by concatenating a sequence of paths $\pi_1, \ldots \pi_n$, where every π_i is a hopping path from $\mu_{fh}(i-1)$ to $\mu_{fh}(i)$. We call paths constructed this way forward hopping paths. The rank of a forward hopping path π_{fh} is the maximum over the ranks of all forward hopping maps from which π_{fh} can be constructed. Similarly we have the notions of backward hopping maps μ_{bh} and backward hopping paths π_{bh}. Given two vertices u, v in G_ρ and a path p from u to v, we define the furthermost vertex $\text{fmv}(p)$ to be the vertex w along p such that for all vertices x in p, $\text{level}(x) \leq \text{level}(w)$ (if there are multiple such occurrences of w, choose the last such occurrence along p). The portion of the path p from u to $\text{fmv}(p)$ is said to be the forward portion and the rest is said to be the backward portion.

Given any regular expression $e \in E$, let A_e denote the minimal DFA that recognizes $\mathcal{L}(e)$ and let n_e denote the number of states in A_e. Now, given a symbolic model ρ and a pair of positions i, j with $j \leq i$, checking if j is a minimal match for e at i is equivalent to checking if the run of A_e on $\rho[j, i]^R$ is accepting and that A_e does not accept any strict prefix of $\rho[j, i]^R$.

The following claim states that a hop cannot be too long. Intuitively, a hop is between two non-τ vertices. The sequence between these two vertices consists of copies of δ and is a minimal match for some regular expression. If there are too many copies of δ, some copies can be removed to get a shorter match for the regular expression.

▷ Claim 10. Suppose $\rho = \tau \cdot \delta^\omega$ is an ultimately periodic consistent symbolic model and $e \in E$ is a regular expression. For all positions $i > |\tau|$, if there exists a match j for e at i such that $j > |\tau|$, then $i - \text{mmp}(i, e) \leq 2n_e|\delta|$.

Non-strict directed cycles in a path do not contribute to its *slen*. So, henceforth when we refer to a path, we mean path without non-strict cycles.

▷ Claim 11. Suppose $\rho = \tau \cdot \delta^\omega$ is an ultimately periodic consistent symbolic model and there are vertices u, v with $slen(u, v) = \omega$. For any $n \in \mathbb{N}$, there is a path from u to v containing either a forward hopping path in its forward portion starting at a vertex whose level is greater than level(u) and level(v), or a backward hopping path in its backward portion starting at a vertex whose level is greater than level(u) and level(v), with the rank of the forward/backward hopping path being at least n.

Proof. The number of long jumps along any $u - v$ path is $\leq 2|V||\tau|$, because every τ-vertex can be visited at most once along a path (since G_ρ has no strict cycles). If the condition in the claim is not satisfied, then there is a position beyond which strict paths cannot go, contradicting the hypothesis that $slen(u, v) = \omega$. ◁

▶ **Definition 12.** *For an ultimately periodic consistent symbolic model $\rho = \tau \cdot \delta^\omega$, let arl (intended to be short for automata run repeat length) be the number $n!|\delta|$, where $n = \max_{e \in E} n_e$.*

Claim 10 proves that hops are short. The next claim proves that long jumps to non-τ vertices can be unboundedly long. Intuitively, such a long jump includes many copies of δ. Some copy may be repeated many times to make the jump longer.

▷ Claim 13. Suppose $\rho = \tau \cdot \delta^\omega$ is an ultimately periodic consistent symbolic model and G_ρ has vertices w_1, w_2 such that $w_1 \cong w_2$, level$(w_1) \geq |\tau| + arl$ and level$(w_2) -$ level(w_1) is a multiple of *arl*. If there is a long jump from w_1 to a τ-vertex y, then there is also a long jump from w_2 to y. Similarly, if there is a long jump from some τ-vertex z to w_2 then there is also a long jump from z to w_1.

Proof. We prove the first case; the proof of the second case is symmetric. Also, we prove for the case where level$(w_2) -$ level$(w_1) = arl$. If level$(w_2) -$ level(w_1) is any multiple of *arl*, we can choose intermediary vertices between w_1 and w_2 with the difference of levels between every pair of consecutive vertices as *arl* and applying the result to every pair of intermediary vertices gives us the required result for w_1 and w_2.

Since there is an edge from w_1 to y, there is a position $k_1 > \max(\text{level}(w_1), \text{level}(y))$ and regular expressions $e_1, e_2 \in E$ such that $\text{mmp}(k_1, e_1) = \text{level}(w_1)$ and $\text{mmp}(k_1, e_2) = \text{level}(y)$.

Consider the position $k_2 = k_1 + arl$. Positions k_1 and k_2 are isomorphic. Therefore, $\text{mmp}(k_2, e_1) = \text{level}(w_2)$. We know $e_1^{-1}\text{var}(w_1) \sim_{k_1} e_2^{-1}\text{var}(y)$, where $\sim \in \{=, <\}$. Also $\text{var}(w_1) = \text{var}(w_2)$ and $\rho(k_1) = \rho(k_2)$. This gives us $e_1^{-1}\text{var}(w_2) \sim_{k_2} e_2^{-1}\text{var}(y)$. So, in order to prove that there is an edge from w_2 to y, it is sufficient to prove that $\text{mmp}(k_2, e_2) = \text{level}(y)$.

Now consider a run of the DFA A_{e_2} on the word $\rho[0, k_2]^R$. There must exist a state s which is visited at at least 2 isomorphic positions along the run. Consider the first such state s visited along the run, let j_1 and j_2 be the first two isomorphic positions at which s is visited. In between positions j_1 and j_2, every state of the automaton could have been visited at most $|\delta|$ times. Therefore, $j_1 - j_2 = m|\delta|$ where $m \leq n_{e_2}$.

By Claim 10, either $\text{mmp}(k_2, e_2) \leq |\tau|$ or $\text{mmp}(k_2, e_2) \geq k_2 - arl$. If $\text{mmp}(k_2, e_2) \geq k_2 - arl$, then due to isomorphism, $\text{mmp}(k_1, e_2) \geq k_1 - arl$, which means that $\text{mmp}(k_1, e_2) > |\tau|$. But this contradicts our assumption that $\text{mmp}(k_1, e_2) = \text{level}(y)$ as y is a τ-vertex. Therefore, $\text{mmp}(k_2, e_2) \leq |\tau|$.

Let $j_1' = j_1 - arl$. Consider the run of A_{e_2} on the word $\rho[0, k_1]^R$. This run visits state s at position j_1' (this is because, we start from the initial state at k_1 and visit the same sequence of states that the run of the word $\rho[0, k_2]^R$ visited between k_2 and j_1).

Along the run of $\rho[0, k_2]^R$, the sequence of states visited between j_1 and j_2 keeps repeating itself after position j_2 (till the first non-τ-vertex) and thus must eventually visit the state s at position j_1'. This is because, $j_1 - j_1' = n!|\delta|$ which is a multiple of $j_1 - j_2 = m|\delta|$ (since $m \leq n_{e_2} \leq n$, m divides $n!$).

We now claim that the first final state visited by the run of $\rho[0, k_2]^R$ must be at level(y). This is because the sequence of states visited beyond position j_1' along the runs of the words $\rho[0, k_1]^R$ and $\rho[0, k_2]^R$ is the same and therefore, we visit a final state at level(y) and a visit to a final state before position level(y) would indicate a shorter minimal match for e_2 at position k_1 contradicting our assumption. Therefore, there is a long jump from w_2 to y. $\lhd$

Next we prove that satisfying $C_{\mathbb{Z}}$ is necessary and sufficient for ultimately periodic consistent symbolic models to admit concrete models. Compared to [12, Lemma 6.2], this proof is more involved due to the presence of long jumps, which are not there in [12].

▶ **Lemma 14** ([12, Lemma 6.2]). *Let $\rho = \tau \cdot \delta^\omega$ be an ultimately periodic consistent symbolic model. Then ρ admits a concrete model over $\mathbb{Z}$ iff ρ satisfies the condition $C_{\mathbb{Z}}$.*

Proof. ($\Rightarrow$) If ρ admits a concrete model over $\mathbb{Z}$ then for all $u, v \in G_\rho$, $slen(u, v) < \omega$. Therefore, condition $C_{\mathbb{Z}}$ is clearly satisfied.

($\Leftarrow$) Let there exist vertices $u, v \in G_\rho$ such that $slen(u, v) = \omega$. Let $l = |V|$. By Claim 11, we can find a path p from u to v such that there is a forward hopping path π_{fh} of rank at least $(2l^2|\delta|^2 + 3) \cdot arl$ starting at a vertex whose level is greater than level(u) and level(v) in the forward portion of p, or a similar backward hopping path. We consider the first case and prove that Condition $C_{\mathbb{Z}}$ is violated, leading to a contradiction. The argument for the second case is symmetric.

Let μ_{fh} be a forward hopping map of rank $(2l^2|\delta|^2 + 3) \cdot arl$ from which π_{fh} can be constructed. Let strict vertices of μ_{fh} be those vertices $\mu_{fh}(i)$ such that the hopping path from $\mu_{fh}(i)$ to $\mu_{fh}(i+1)$ (along π_{fh}) is a strict hopping path. One level may contain at most one strict vertex of μ_{fh}. So, μ_{fh} must contain strict vertices in at least $(2l^2|\delta|^2 + 3) \cdot arl$ distinct levels. Suppose i_{min} and i_{max} are the minimum and maximum among these levels. Take the subgraph of G_ρ induced by vertices in levels between i_{min} and i_{max} and divide it into $\lfloor \frac{i_{max} - i_{min}}{arl} \rfloor$ portions containing arl levels each and a last portion containing the remaining levels ($< arl$). At least $(2l^2|\delta|^2 + 3)$ portions contain strict vertices of μ_{fh}.

We look at the forward hopping path after the first portion. Now we drop alternate portions and from each of the remaining $(l^2|\delta|^2 + 1)$ portions, take one strict vertex so that the difference in levels between picked vertices is at least arl. Among these vertices, we can find a set of at least $l|\delta| + 1$ vertices, all belonging to the same isomorphism class (since there are $l|\delta|$ isomorphism classes). Let the vertices in this set be $w_1, w_2, \cdots w_{l|\delta|+1}$ (in increasing order of their levels). Note that there is a strict path from w_i to w_{i+1} for all $i \in [1, l|\delta|]$. Let us now look at the portion of the path from $w_{l|\delta|+1}$ to v. There can be two possible cases:

Case a. The portion of the path from $w_{l|\delta|+1}$ to the first vertex occurring after $w_{l|\delta|+1}$ along the path whose level is less than level(w_1), is a hopping path.

We know by Claim 10 that the difference in levels between the end-points of a hop must be less than arl. Hence, along this hopping path, a set of vertices $w_1', w_2' \cdots w_{l|\delta|+1}'$ must be visited such that $|\text{level}(w_i) - \text{level}(w_i')| < arl$ for all $i \in [1, l|\delta| + 1]$. Among these $l|\delta| + 1$ vertices, at least two must be isomorphic. Let us say (w_i', w_j') is such a pair of isomorphic vertices along this hopping path with $i < j$ (see the picture on the left side of Fig. 2).

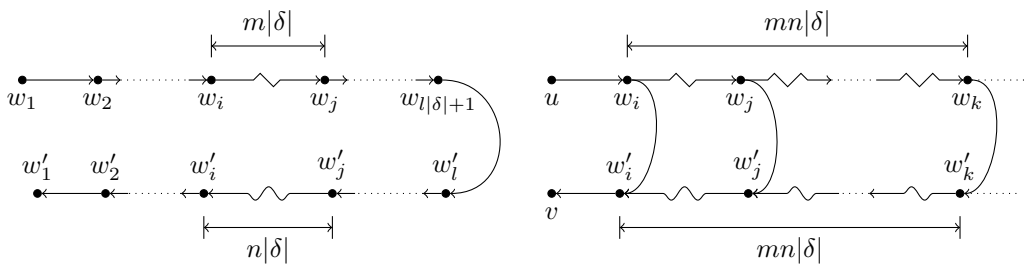


Figure 2 Illustration for **case a** of the proof of Lemma 14.

Now, let $|\text{level}(w_j) - \text{level}(w_i)| = m|\delta|$ and $|\text{level}(w'_j) - \text{level}(w'_i)| = n|\delta|$ for some $m, n \in \mathbb{N}$. Note that $|\text{level}(w_i) - \text{level}(w'_i)|$ and $|\text{level}(w_j) - \text{level}(w'_j)|$ need not be equal. But since $w'_i \cong w'_j$ and $w_i \cong w_j$, it is possible to pump the portion of the path from w'_i to w'_j m times and the portion of the path from w_i to w_j n times, resulting in a forward path from u to some vertex w_k passing through w_i and in a backward path from v to some vertex w'_k passing through w'_i such that $w'_i \cong w'_k$, $w_i \cong w_k$ and $|\text{level}(w_i) - \text{level}(w'_i)| = |\text{level}(w_k) - \text{level}(w'_k)|$. This is illustrated in the picture on the right side of Fig. 2. Since there is a directed path from w_i to w'_i that does not visit any τ-vertex, there must also be a directed path from w_k to w'_k (a path isomorphic to that from w_i to w'_i). Note that the portion of the path from w_i to w_k along p is strict. It is therefore possible to pump the portion of the forward and backward paths from w_i to w_k and respectively from w'_i to w'_k infinitely often, to obtain a forward $u - v$ witness. This results in a violation of condition $C_{\mathbb{Z}}$, giving rise to a contradiction.

Case b. In the portion of the path from $w_{l|\delta|+1}$ to v, before a vertex with level less than $\text{level}(w_1)$ is visited, there is a long jump.

Let the first long jump along the portion of the path from $w_{l|\delta|+1}$ to v be from a vertex z to a τ-vertex y. Figure 3 illustrates this. Consider a vertex $w_{l|\delta|+2}$ isomorphic to $w_{l|\delta|+1}$ such that $(\text{level}(w_{l|\delta|+2}) - \text{level}(w_{l|\delta|+1})) = (\text{level}(w_{l|\delta|+1}) - \text{level}(w_{l|\delta|})) \cdot arl$. Since $(\text{level}(w_{l|\delta|+2}) - \text{level}(w_{l|\delta|+1}))$ is a multiple of $(\text{level}(w_{l|\delta|+1}) - \text{level}(w_{l|\delta|}))$ and since $w_{l|\delta|+1} \cong w_{l|\delta|+2}$, there must be a strict path from $w_{l|\delta|+1}$ to $w_{l|\delta|+2}$ as well. We now "mimic" the

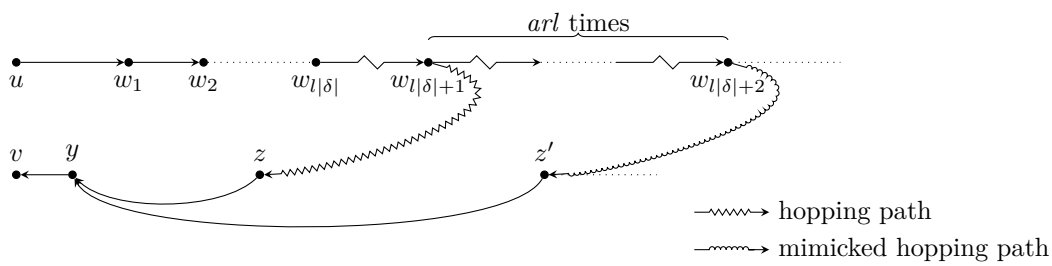


Figure 3 Illustration for **case b** of the proof of Lemma 14.

path from $w_{l|\delta|+1}$ to z by taking an isomorphic copy of that path from $w_{l|\delta|+2}$. We have $(\text{level}(w_{l|\delta|+2}) - \text{level}(w_{l|\delta|+1})) = (\text{level}(w_{l|\delta|+1}) - \text{level}(w_{l|\delta|})) \cdot arl$ and the portion of the path from $w_{l|\delta|+1}$ to z consists only of hops. So we basically construct a path of hops from $w_{l|\delta|+2}$ such that the level of every vertex along that path is exactly $(\text{level}(w_{l|\delta|+1}) - \text{level}(w_{l|\delta|})) \cdot arl$ more than the level of the corresponding vertex in the path from $w_{l|\delta|+1}$ to z. This means that corresponding to position z is the position z' in this new path such that $(\text{level}(z') - \text{level}(z)) = (\text{level}(w_{l|\delta|+1}) - \text{level}(w_{l|\delta|})) \cdot arl$. By construction, we know that $\text{level}(z) > |\tau| + arl$. Therefore

by Claim 13, there exists a long jump from z' to y, hence there exists a path from $w_{l|\delta|+2}$ to v. We can now repeat this process by considering a vertex $w_{l|\delta|+3}$ which is isomorphic to $w_{l|\delta|+2}$ and whose level is exactly $(\text{level}(w_{l|\delta|+1}) - \text{level}(w_{l|\delta|})) \cdot arl$ more than that of $w_{l|\delta|+2}$. Now, we mimic the path from $w_{l|\delta|+2}$ to z' as before, to get a path from $w_{l|\delta|+3}$ to v. The paths from $w_{l|\delta|+i}$ to $w_{l|\delta|+i+1}$ is strict for all $i \in \mathbb{N}$ by construction. Thus, we repeat this process infinitely many times to get a forward $u - v$ witness with the sequence of vertices $w_{l|\delta|+1}, w_{l|\delta|+2}, w_{l|\delta|+3} \cdots$ being the infinitely many vertices along the witness such that there is a directed path from each of them to v. This again leads to a violation of condition $C_{\mathbb{Z}}$, contradicting our assumption.

In the case where there is a backward hopping path in the backward portion of p, a symmetric argument gives us a backward $u - v$ witness, violating condition $C_{\mathbb{Z}}$. ◀

We shall now define an MSO formula $\psi_{C_{\mathbb{Z}}}$ whose language is the set of all symbolic models that satisfy condition $C_{\mathbb{Z}}$. The formula $succ(x, y, Z)$ holds iff the position y is a successor of position x when restricted to the set Z.

$$\text{succ}(x, y, Z) \equiv ((x < y) \wedge (x \in Z) \wedge (y \in Z) \wedge \forall z((x < z < y) \implies \neg(z \in Z)))$$

The formula $\text{inf}(Z)$ holds iff there are infinitely many positions in Z.

$$\text{inf}(Z) \equiv \forall y \exists z((z > y) \wedge (z \in Z))$$

The formula $\text{sfp}(Z, Z_1, \ldots, Z_l)$ (strict forward path) holds iff $Z = \bigcup_{i=1}^{l} Z_i$ and between every pair of consecutive positions x, y in Z, if $x \in Z_p$ and $y \in Z_{p'}$, then there is a strict directed path from the vertex (v_p, x) to $(v_{p'}, y)$ in G_ρ. The formula $\text{sbp}(Z, Z_1, \ldots, Z_l)$ (strict backward path) is symmetric.

$$\begin{aligned}\text{sfp}(Z, Z_1, \ldots, Z_l) \equiv & \forall z(\bigvee_{i=1}^{l}(z \in Z_i) \Leftrightarrow z \in Z) \wedge \\ & \forall x \forall y(\bigwedge_{p,p' \in [1,l]}((x \in X_p \wedge y \in X_{p'} \wedge succ(x, y, Z)) \implies \text{path}^{<}_{p,p'}(x, y))\end{aligned}$$

$$\begin{aligned}\text{sbp}(Z, Z_1, \ldots, Z_l) \equiv & \forall z(\bigvee_{i=1}^{l}(z \in Z_i) \Leftrightarrow z \in Z) \wedge \\ & \forall x \forall y(\bigwedge_{p,p' \in [1,l]}((x \in X_p \wedge y \in X_{p'} \wedge succ(x, y, Z)) \implies \text{path}^{<}_{p',p}(y, x))\end{aligned}$$

The formula $\psi^a_{C_{\mathbb{Z}}}$ (resp. $\psi^b_{C_{\mathbb{Z}}}$) denotes that there do not exist vertices (v_i, u) and (v_j, v) in G_ρ such that there is a forward (resp. backward) $(v_i, u) - (v_j, v)$ witness.

$$\begin{aligned}\psi^a_{C_{\mathbb{Z}}} \equiv \neg(\exists u \exists v \bigvee_{i,j \in [1,l]} & (\exists X \exists X_1 \ldots \exists X_l \wedge ((\text{sfp}(X, X_1, \ldots, X_l) \wedge \text{inf(X)} \wedge (u \in X_i) \\ & \wedge \forall y(\bigwedge_{k \in [1,l]}((y \in X_k) \implies \text{path}^{\leq}_{k,j}(y, v))))\end{aligned}$$

$$\begin{aligned}\psi^b_{C_{\mathbb{Z}}} \equiv \neg(\exists u \exists v \bigvee_{i,j \in [1,l]} & (\exists X \exists X_1 \ldots \exists X_l((\text{sbp}(X, X_1, \ldots, X_l) \wedge \text{inf(X)} \wedge (v \in X_j) \\ & \wedge \forall y(\bigwedge_{k \in [1,l]}((y \in X_k) \implies \text{path}^{\leq}_{i,k}(u, y))))\end{aligned}$$

Now we define the formula $\psi_{C_\mathbb{Z}}$ as $\psi_{C_\mathbb{Z}} \equiv \psi^a_{C_\mathbb{Z}} \vee \psi^b_{C_\mathbb{Z}}$. It is easy to verify that $\psi_{C_\mathbb{Z}}$ is true iff the word satisfies condition $C_\mathbb{Z}$.

As discussed earlier, the language $\mathcal{L}^{sat}_\phi$ (described in Section 4) is ω-regular, hence by Büchi's theorem, we can define an MSO formula ψ^{sat}_ϕ whose language is $\mathcal{L}^{sat}_\phi$. ψ_{symb}, which we have defined in the previous section, is the MSO formula whose language is $\mathcal{L}_{symb}$. Now let ϕ be the given RCLTL($\mathcal{Z}$) formula. We define the MSO formula $\psi^\mathcal{Z}_\phi = \psi^{sat}_\phi \wedge \psi_{symb} \wedge \psi_{C_\mathbb{Z}}$. We now have the following lemma:

▶ **Lemma 15.** *An RCLTL($\mathcal{Z}$) formula ϕ is satisfiable iff $\mathcal{L}(\psi^\mathcal{Z}_\phi)$ is non-empty.*

Proof. ($\Rightarrow$) Suppose ϕ is satisfiable. Let σ be a concrete model such that $\sigma \models \phi$. Let $\rho = \mu(\sigma)$. By Lemma 2, $\rho \in \mathcal{L}(\psi^{sat}_\phi)$. By definition, ρ is a consistent symbolic model, hence $\rho \in \mathcal{L}(\psi_{symb})$. Also since ρ admits a concrete model over $\mathbb{Z}$, by Lemma 9, ρ satisfies condition $C_\mathbb{Z}$, hence $\rho \in \mathcal{L}(\psi_{C_\mathbb{Z}})$. Thus $\rho \in \mathcal{L}(\psi^{sat}_\phi) \cap \mathcal{L}(\psi_{symb}) \cap \mathcal{L}(\psi_{C_\mathbb{Z}})$. Hence $\rho \in \mathcal{L}(\psi^\mathcal{Z}_\phi)$.

($\Leftarrow$) Suppose a word ρ satisfies the formula $\psi^\mathcal{Z}_\phi$. Then, there must exist an ultimately periodic word ρ' such that ρ' satisfies $\psi^\mathcal{Z}_\phi$. Since $\rho' \in \mathcal{L}(\psi^\mathcal{Z}_\phi)$, ρ' is a symbolic model that symbolically satisfies ϕ and also satisfies condition $C_\mathbb{Z}$. By Lemma 14, ρ' admits a concrete model σ over $\mathbb{Z}$. Also, since $\rho' \models_s \phi$, by Lemma 2, $\sigma \models \phi$. Thus, ϕ is RCLTL($\mathcal{Z}$)-satisfiable. ◀

▶ **Theorem 16.** *The satisfiability problem for RCLTL($\mathcal{Z}$) is decidable.*

Proof. Given an RCLTL($\mathcal{Z}$) formula ϕ, we can effectively convert the MSO formula to a Büchi automaton $\mathcal{A}^\mathcal{Z}_\phi$ using the construction of Büchi's theorem [5] and check it for non-emptiness. ◀

6 Discussion and Future Work

In this paper, we consider an extension RCLTL of Constraint LTL to let atomic formulas access positions that are unboundedly far away in the past. We prove that the satisfiability problem for RCLTL is decidable over dense and open domains, and also over the integers with linear order and equality. With some changes in the technical details of the proofs, we think that the satisfiability problem can be proven to be decidable if we allow the atomic formulas to access positions that are unboundedly far away in the future as well, and also over the domain of natural numbers.

Based on the quantifier alternations in the MSO formulas that we have used, a rough complexity upper bound of six exponentials may be worked out. Figuring out the exact complexity of the satisfiability problem is left for future work.

References

1. Marcello M Bersani, Domenico Bianculli, Schahram Dustdar, Alessio Gambi, Carlo Ghezzi, and Srđan Krstić. Towards the formalization of properties of cloud-based elastic systems. In *proceedings of the 6th international workshop on principles of engineering service-oriented and cloud systems*, pages 38–47, 2014.
2. Marcello M Bersani, Matteo Rossi, and Pierluigi San Pietro. A tool for deciding the satisfiability of continuous-time metric temporal logic. *Acta Informatica*, 53(2):171–206, 2016.
3. Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*, 12(4):1–26, 2011.
4. Ahmed Bouajjani and Peter Habermehl. Constrained properties, semilinear systems, and petri nets. In *International Conference on Concurrency Theory*, pages 481–497. Springer, 1996.

5 J Richard Buchi. On a decision method in restricted second order arithmetic. *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science, 1960*, 1960.

6 Hubert Comon and Véronique Cortier. Flatness is not a weakness. In *International workshop on computer science logic*, pages 262–276. Springer, 2000.

7 Stephane Demri, Deepak D'Souza, and Régis Gascon. Temporal logics of repeating values. *Journal of Logic and Computation*, 22, October 2012. doi:10.1093/logcom/exr013.

8 Stéphane Demri and Régis Gascon. The effects of bounding syntactic resources on presburger ltl. *Journal of Logic and Computation*, 19(6):1541–1575, 2009.

9 Stéphane Demri and Ranko Lazić. Ltl with the freeze quantifier and register automata. *ACM Trans. Comput. Logic*, 10(3), April 2009. doi:10.1145/1507244.1507246.

10 Stéphane Demri and Karin Quaas. Concrete domains in logics: a survey. *ACM SIGLOG News*, 8(3):6–29, 2021.

11 Stéphane Demri and Karin Quaas. Constraint automata on infinite data trees: from ctl(z)/ctl*(z) to decision procedures. In *34th International Conference on Concurrency Theory (CONCUR 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CONCUR.2023.29.

12 Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint ltl. *Information and Computation*, 205(3):380–415, 2007. doi:10.1016/j.ic.2006.09.006.

13 Cindy Eisner and Dana Fisman. Functional specification of hardware via temporal logic. *Handbook of Model Checking*, pages 795–829, 2018.

14 Bernd Finkbeiner, Philippe Heim, and Noemi Passing. Temporal stream logic modulo theories. In *International Conference on Foundations of Software Science and Computation Structures*, pages 325–346. Springer International Publishing Cham, 2022.

15 Bernd Finkbeiner, Felix Klein, Ruzica Piskac, and Mark Santolucito. Temporal stream logic: Synthesis beyond the bools. In *International Conference on Computer Aided Verification*, pages 609–629. Springer, 2019.

16 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.

17 Nadia Labai, Magdalena Ortiz, and Mantas Šimkus. An exptime upper bound for alc with integers. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 17, pages 614–623, 2020.

18 Gabor Madl, Luis Bathen, German Flores, and Divyesh Jadav. Formal verification of smart contracts using interface automata. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 556–563, 2019. doi:10.1109/Blockchain.2019.00081.

19 M Praveen, Diego Figueira, and Stephane Demri. Reasoning about data repetitions with counter systems. *Logical Methods in Computer Science*, 12, 2016.

20 Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

21 Moshe Y Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 322–331. IEEE Computer Society, 1986.

A Details of Section 3

Proof of Lemma 5. We shall prove the lemma by showing that the graph G_ρ admits an edge-respecting labelling $l\colon V \times \mathbb{N} \to D$.

We assume an ordering $\prec$ on V and use it to define an ordering on the vertices of G_ρ given by $(x, i) \prec (y, j)$ iff $i < j$, or $i = j$ and $x \prec y$. We now give a procedure to label the vertices.

1. Label the vertices in order of $\prec$. Begin by labelling the first, say $(x, 0)$ by any element $d \in D$.
2. In general, let X be the portion of the graph already labelled, and u be the next vertex to be labelled. Let $X_{=u}$ denote the set of vertices in X to which there is an $=$-labelled path from u; $X_{<u}$ denote the set of vertices in X from which there is a strict directed path to u; and $X_{u<}$ denote the set of vertices in X to which there is a strict directed path from u. Let $p = \max\{l(v) \mid v \in X_{<u}\}$ and $q = \min\{l(w) \mid w \in X_{u<}\}$ (we will use the values of p and q only when the sets $X_{<u}$ and $X_{u<}$ are, respectively, non-empty). The values p and q are in D, which is dense and open. So there exist $p', q', r \in D$ such that $p < p'$, $q' < q$ and $p < r < q$. Now we can label u as follows:
   - **(a)** if $X_{=u}$ is non-empty, label u by $l(v)$ for any $v \in X_{=u}$
   - **(b)** if $X_{<u}$ is non-empty and both $X_{=u}$ and $X_{u<}$ are empty, label u by p'.
   - **(c)** if $X_{u<}$ is non-empty and both $X_{=u}$ and $X_{<u}$ are empty, label u by q'.
   - **(d)** if $X_{<u}$ and $X_{u<}$ are non-empty and $X_{=u}$ is empty, label u by r.
   - **(e)** if $X_{<u}$, $X_{u<}$ and $X_{=u}$, all three are empty, label u by any arbitrary element d of D.

We now aim to prove that the labelling l that we obtain for the vertices of the graph G_ρ is edge-respecting. In order to prove this, we state the following invariant and we shall prove inductively that the invariant holds at the end of every iteration of the procedure.

Invariant: The labelling when restricted to the subgraph X is edge-respecting. Base case: X is the graph with just one vertex $(x, 0)$. In this case, any labelling for $(x, 0)$ is trivially edge-respecting.

Inductive Step: Let X be the portion of the graph labelled till now. By Induction hypothesis, the labelling of all vertices in X is edge-respecting. Now, let u be the vertex that is next labelled by the procedure. We wish to prove that the label $l(u)$ is such that the labelling l is an edge-respecting labelling of the graph $X \cup \{u\}$.

To argue that $l(u)$ is a valid edge-respecting label for u, suppose to the contrary, that it was not. Then there must exist a vertex v in $X \cup \{u\}$ and a directed path from u to v or from v to u which is inconsistent with the labels of u and v. First note that such a v cannot be equal to u. This is because, we know that there is no strict cycle in the graph and any path from u to itself must contain only $=$-labelled edges. The labelling of u is therefore, trivially consistent with every path from u to itself. Thus, the vertex v must be from X.

Observe now that if u was labelled by clause 2b, 2c, 2d, or 2e, then $l(u)$ must clearly be a valid label. If, for instance, u was labelled by clause 2b, then it must be the case that there is no directed path from u to any vertex in X. u could have been labelled inconsistently only if there is a vertex v in X, such that there is a strict directed path from v to u but $l(v) \geq l(u)$. But since u was labelled using clause 2b, $l(u) = p' > \max\{l(v) \mid v \in X_{<u}\}$. Therefore $l(u)$ must be $> l(v)$. Using a similar argument, it is possible to see that even if u was labelled by clauses 2c, 2d, or 2e, $l(u)$ must be a valid label.

So it must be the case that u was labelled by clause 2a. u could have been labelled inconsistently in the following three possible cases: there exists a strict directed path from u to v but $l(u) \geq l(v)$ or there exists a strict directed path from v to u but $l(u) \leq l(v)$ or there exists a directed path from u to v with all $=$-labelled edges but $l(u) \neq l(v)$. The first two cases are symmetric (and hence arguing for the first and third cases would suffice).

The first case can arise only if there is another vertex w in X such that there is an $=$-labelled path from w to u, and $l(u) = l(w)$. This gives us a strict directed path from w to v. This also means that $l(w) \geq l(v)$. But since l is an edge-respecting labelling for X, $l(w)$ must be $< l(v)$ giving rise to a contradiction. Hence, this case is not possible.

The third case, again, could only have occurred if u was labelled by 2a. Thus, there is a vertex w such that there is an $=$-labelled path from u to w and $l(u) = l(w)$. This now means that there is a directed path with all $=$-labelled edges from w to v (hence, also from v to w) but with $l(w) \neq l(v)$. Again, since l is an edge-respecting labelling for X, $l(w)$ must be equal to $l(v)$, giving rise to a contradiction.

This proves that the label l is an edge-respecting labelling for $X \cup \{u\}$. Thus, the invariant holds at the end of every iteration of the procedure. Hence the labelling l obtained is indeed an edge-respecting labelling for the graph G_ρ. Thus, every symbolic model ρ with all memoirs relevant and whose associated graph G_ρ has no strict cycle, admits a concrete model over D. ◀

B Details of Section 4

▷ Claim 17. Consider the function $f_x^i \colon (2^{\mathbb{N}})^l \to (2^{\mathbb{N}})^l$ given by: $f_x^i(X_1, X_2, \ldots, X_l) = (X_1', X_2', \ldots, X_l')$ where $X_k' = \{y \mid \sim_{i,k} (x, y)\} \cup \{y \mid \exists k', \exists z \in X_{k'} \text{ such that } \sim_{k',k} (z, y)\} \cup X_k$. We claim that the function f_x^i has a least fixed-point.

Proof. Consider the partial ordering $\leq$ on $(2^{\mathbb{N}})^l$, given by $(X_1, \ldots, X_l) \leq (Y_1, \ldots, Y_l)$ if for all $i \in [1, l]$, $X_i \subseteq Y_i$. Notice that all subsets of $(2^{\mathbb{N}})^l$ have both a greatest lower bound and a least upper bound and thus, $((2^{\mathbb{N}})^l, \leq)$ forms a complete lattice. Also note that the function f_x^i as defined above is non-decreasing, hence a monotonic function. Thus, by Knaster-Tarski theorem [20], the set of fixed-points of $(2^{\mathbb{N}})^l$ forms a complete lattice under $\leq$. This guarantees the existence of a least fixed-point of f_x^i. ◁

▷ Claim 18. $\text{closed}_i^{\sim}(x, Y_1, \ldots, Y_l)$ holds iff $(Y_1, \ldots, Y_l)$ is a fixed-point of f_x^i.

Proof. $(Y_1, \ldots, Y_l)$ is a fixed-point of f_x^i

$\Leftrightarrow f_x^i((Y_1, \ldots Y_l)) = (Y_1, \ldots, Y_l)$.

$\Leftrightarrow \forall k \in [1, l], Y_k \cup \{y \mid \sim_{i,k} (x, y)\} \cup \{y \mid \exists k', \exists z \in X_{k'} \text{ such that } \sim_{k',k} (z, y)\} = Y_k$.

$\Leftrightarrow \forall k \in [1, l], \forall z$ if $\exists k', \exists x' \in Y_{k'}$ such that if one of $\sim_{k,k'} (x', z)$ or $\sim_{i,k} (x, z)$ holds, then, $z \in Y_{k'}$.

$\Leftrightarrow \forall x' \forall z (x' \in Y_{k'} \wedge \sim_{k',k} (x', z) \implies z \in Y_k) \wedge \forall z \bigwedge_{k \in [1,l]} (\sim_{i,k} (x, z) \implies z \in Y_k)$

$\Leftrightarrow \text{closed}_i^{\sim}(x, Y_1, \ldots, Y_l)$ holds ◁

▷ Claim 19. If $X_1^n, \ldots X_l^n$ is obtained upon n iterations of f_x^i to $(\emptyset, \ldots, \emptyset)$, and if $z \in X_k^n$, then there is a path of length $\leq n$ (but non-zero) from (v_i, x) to (v_k, z) in G_ρ.

Proof. Let us prove this by induction on n. Take $n = 1$ as the base case. By definition, $f_x^i((\emptyset, \ldots, \emptyset))$ gives us the tuple $(y_1, \ldots, y_l)$ such that there is an edge from (v_i, x) to (v_j, y_j) for all $j \in [1, l]$. Clearly, the hypothesis holds in the base case.

Inductive step: Suppose the inductive hypothesis is true for all $m < n$. Let $X_1^n, \dots X_l^n$ be obtained upon n iterations of f_x^i to $(\emptyset, \dots, \emptyset)$. Now by induction hypothesis, there is a path of length $\leq (n-1)$ from (v_i, x) to (v_k, z) for all $k \in [1, l]$, for all $z \in X_k^{n-1}$. Now consider $z \in X_k^n \setminus X_k^{n-1}$. By definition, it must be the case that there exists $k' \in [1, l]$ and $z' \in X_{k'}^{n-1}$ such that $\sim_{k',k} (z', z)$ holds. Now there is a path of length $\leq (n-1)$ from (v_i, x) to $(v_{k'}, z')$, hence there is a path of length $\leq n$ from (v_i, x) to (v_k, z). ◁

▷ Claim 20. $\text{path}_{i,j}^{\sim}(x, y)$ holds iff there is a path of non-zero length from (v_i, x) to (v_j, y).

Proof. ($\Leftarrow$) We prove this by an induction on the length of the path. For the base case, assume there is a path of length 1, that is, $\sim_{i,j} (x, y)$ holds. Now by definition of the formula $\text{closed}_i^{\sim}(x, Y_1, \dots, Y_l)$, and since $\sim_{i,j} (x, y)$, for all $Y_1, \dots, Y_l$, if $\text{closed}_i^{\sim}(x, Y_1, \dots, Y_l)$ holds then $y \in Y_j$ must hold. Or in other words, the formula $\text{path}_{i,j}^{\sim}(x, y)$ holds.

Inductive step: Let us assume that there is a path of length n from (v_i, x) to (v_j, y). This means that there exists (v_k, z) such that there is a path of length $(n-1)$ from (v_i, x) to (v_k, z) and such that $\sim_{k,j} (z, y)$ holds. By induction hypothesis, this means that, both, $\text{path}_{i,k}^{\sim}(x, z)$ and $\sim_{k,j} (z, y)$ hold. By the definitions of $\text{path}_{i,k}^{\sim}(x, z)$ and $\text{closed}_i^{\sim}(x, Y_1, \dots, Y_l)$, we get that $\forall Y_1 \dots \forall Y_l (\text{closed}_i^{\sim}(x, Y_1, \dots, Y_l) \implies y \in Y_j)$ holds, or in other words, $\text{path}_{i,j}^{\sim}(x, y)$ holds.

($\Rightarrow$) We assume that $\text{path}_{i,j}^{\sim}(x, y)$ holds. This means, using Claim 18, that if $(Y_1, \dots, Y_l)$ is a fixed-point, then $y \in Y_j$. In particular, if $(Y_1, \dots, Y_l)$ is the least fixed-point, then $y \in Y_j$. We first prove that the least fixed-point of f_x^i can be obtained by iteratively applying f_x^i on $(\emptyset, \dots, \emptyset) \leq \omega$ times. In order to prove this, it is sufficient to prove that $(f_x^i)^{\omega+1}((\emptyset, \dots, \emptyset)) = (f_x^i)^{\omega}((\emptyset, \dots, \emptyset))$. By definition, $(f_x^i)^{\omega+1}((\emptyset, \dots, \emptyset)) = f_x^i((f_x^i)^{\omega}((\emptyset, \dots, \emptyset)))$ and $(f_x^i)^{\omega}((\emptyset, \dots, \emptyset)) = \bigcup_{n=0}^{\infty} (f_x^i)^n((\emptyset, \dots, \emptyset))$. Let us assume $(f_x^i)^{\omega}((\emptyset, \dots, \emptyset)) = (X_1, \dots X_l)$ and let $f_x^i((X_1, \dots X_l)) = (X_1', \dots X_l')$. Suppose for some $k \in [1, l]$, $X_k \subset X_k'$. This means that there exists $y \in X_k' \setminus X_k$ such that for some $k' \in [1, l]$ and for some $z \in X_{k'}$, $\sim_{k',k} (z, y)$ holds. But since, $X_{k'}$ contains z, the position z must have been added to the k'^{th} component of the l-tuple, in some n^{th} iteration of the function f_x^i for some $n \in \mathbb{N}$. This means that y should have been added to the kth component of the l-tuple, in the $(n+1)^{\text{th}}$ iteration of f_x^i contradicting our assumption that $y \in X_k' \setminus X_k$. This means that $X_k' = X_k$ for all $k \in [1, l]$ implying that $(f_x^i)^{\omega+1}((\emptyset, \dots, \emptyset)) = (f_x^i)^{\omega}((\emptyset, \dots, \emptyset))$.

Now, if the least fixed-point of f_x^i can be obtained by applying f_x^i n times on $(\emptyset, \dots, \emptyset)$ for some $n \in \mathbb{N}$, then by Claim 19, there exists a path of length $\leq n$ from (v_i, x) to (v_j, y). Now, if the least fixed-point of f_x^i is obtained by applying f_x^i ω times on $(\emptyset, \dots, \emptyset)$, we need to perform a transfinite induction in order to prove that there is still a path from (v_i, x) to (v_j, y) in this case. Let $(f_x^i)^{\omega}((\emptyset, \dots, \emptyset)) = (Y_1, \dots, Y_l)$ be the least fixed-point. We know $y \in Y_j$. Now by definition of $(f_x^i)^{\omega}$, the element y must have been added to the kth component of the l-tuple in the nth iteration of f_x^i for some $n \in \mathbb{N}$. But then, this means that there is a path of length $\leq n$ from (v_i, x) to (v_j, y). Thus, there is a path of non-zero length from (v_i, x) to (v_j, y) even in this case, as required. ◁

C Details of Section 5

Proof of Lemma 9. ($\Rightarrow$) ρ admits some concrete model σ. Let l be the corresponding edge-respecting labelling of the graph G_ρ. Then, clearly, $slen(u, v) \leq |l(v) - l(u)|$, for all vertices $u, v \in G_\rho$. Thus, there cannot exist vertices u and v with $slen(u, v) = \omega$.

($\Leftarrow$) Suppose G_ρ satisfies the given condition. We outline a procedure that gives an edge-respecting $\mathbb{Z}$-labelling $l \colon V \times \mathbb{N} \to \mathbb{Z}$ of G_ρ.

We assume an ordering $\prec$ on V and use it to define an ordering on the vertices of G_ρ given by $(x, i) \prec (y, j)$ iff $i < j$, or $i = j$ and $x \prec y$. We now give a procedure to label the vertices.

1. Label the vertices in order of $\prec$. Begin by labelling the first, say $(x, 0)$ by 0.
2. In general, let X be the portion of the graph already labelled, and u be the next vertex to be labelled. Let $X_{\leq u}$ denote the set of vertices in X from which there is a directed path to u and $X_{u\leq}$ denote the set of vertices in X to which there is a directed path from u. Now we can label u as follows:
   - **(a)** if $X_{u\leq}$ is non-empty then set $l(u) = \min\{(l(v) - slen(u, v)) \mid v \in X, \exists$ a path from u to $v\}$, else,
   - **(b)** if $X_{\leq u}$ is non-empty, set $l(u) = \max\{(l(v) + slen(v, u)) \mid v \in X, \exists$ a path from v to $u\}$, else
   - **(c)** if both $X_{\leq u}$ and $X_{u\leq}$ are empty, label u by 0.

We now aim to prove that the labelling l that we obtain for the vertices of the graph G_ρ is edge-respecting. In order to prove this, we state the following invariant and we shall prove inductively that the invariant holds at the end of every iteration of the procedure.

Invariant: The labelling when restricted to the subgraph X is edge-respecting. Base case: X is the graph with just one vertex $(x, 0)$. In this case, any labelling for $(x, 0)$ is trivially edge-respecting.

Inductive Step: Let X be the portion of the graph labelled till now. By Induction hypothesis, the labelling of all vertices in X is edge-respecting. Now, let u be the vertex that is next labelled by the procedure. We wish to prove that the label $l(u)$ is such that the labelling l is an edge-respecting labelling of the graph $X \cup \{u\}$.

To argue that $l(u)$ is an valid edge-respecting label for u, suppose to the contrary, that it was not. Then there must exist a vertex v in $X \cup \{u\}$ and a directed path from u to v or from v to u which is inconsistent with the labels of u and v. First note that such a v cannot be equal to u. This is because, we know that there is no strict cycle in the graph and any path from u to itself must contain only $=$-labelled edges. The labelling of u is therefore trivially consistent with every path from u to itself. Thus the vertex v must be from X.

Therefore, u could have been labelled inconsistently in the following two possible cases: there exists a directed path p from u to v but $slen(p) > l(v) - l(u)$ or there exists a directed path p from v to u but $slen(p) > l(u) - l(v)$.

For the first case to arise it must be the case that u was labelled by clause 2a. But then the label $l(u)$ must be such that $l(u) \leq l(v) - slen(p)$. Thus, this case is not possible.

There are two possibilities for the second case. In the first possibility, u was labelled by clause 2a. So there must have been a vertex w in X with a path q from u to w, and $l(u) = l(w) - slen(q)$. But since v and w were labelled without any discrepancy, it must be the case that $l(w) - l(v) \geq slen(p) + slen(q)$. Thus, $l(u) = l(w) - slen(q) \geq l(v) + slen(p)$. This contradicts our assumption that $slen(p) > l(u) - l(v)$, and hence this possibility is ruled out. The second possibility was that u was labelled by an application of clause 2b of the procedure. But then it must be the case that $l(u) \geq l(v) + slen(p)$. Thus, this case is also ruled out.

This proves that the label l is an edge-respecting labelling for $X \cup \{u\}$. Thus, the invariant holds at the end of every iteration of the procedure and hence, the labelling l obtained is indeed an edge-respecting labelling for the graph G_ρ. Hence, every consistent symbolic model ρ satisfying $slen(u, v) < \omega$ for all $u, v \in G_\rho$, admits a concrete model over $\mathbb{Z}$. ◀

Proof of Claim 10. There can be two cases.
Case 1: $i - j \leq n_e(|\delta| + 1)$.

In this case, $(i - mmp(i, e)) \leq (i - j) < 2n_e|\delta|$.

Case 2: $i - j > n_e(|\delta| + 1)$.

Consider a run of A_e on the word $\rho[j, i]^R$. Now since, $i - j > n_e(|\delta| + 1)$, at least one state s of A_e must be visited at a pair of isomorphic positions along this run. Consider the first state s along the run, that is visited at a pair of isomorphic positions. Let k_1 and k_2 be the first pair of isomorphic positions at which s is visited. Consider the portion of the run from i to k_1. Each state of A_e can be visited at most $|\delta|$ times in this portion, hence $i - k_1 \leq n_e|\delta|$. Now since k_1 and k_2 are isomorphic positions, $k_1 - k_2 = m|\delta|$ for some $m \in \mathbb{N}$.

Also, since between positions k_1 and k_2, every state of A_e can be visited at most $|\delta|$ times, $m \leq n_e$.

Now the sequence of states visited between positions k_1 and k_2 repeats itself along the run until the last time the state s is visited at a position isomorphic to k_1 along the run. Let k_l be the last position isomorphic to k_1, at which s is visited. We know a final state F is visited at position j. Now, since k_1 and k_l are isomorphic, the final state F must have been visited at position $k_1 - (k_l - j) = k_1 - k_l + j$. This means that $\text{mmp}(i, e) = k_1 - k_l + j$. Therefore, $i - \text{mmp}(i, e) = (i - k_1) + (k_l - j) \leq 2n_e|\delta|$. Hence, proved. $\triangleleft$

Counter Machines with Infrequent Reversals

Alain Finkel
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, Gif-sur-Yvette, France

Shankara Narayanan Krishna
IIT Bombay, India

Khushraj Madnani
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Rupak Majumdar
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

Bounding the number of reversals in a counter machine is one of the most prominent restrictions to achieve decidability of the reachability problem. Given this success, we explore whether this notion can be relaxed while retaining decidability.

To this end, we introduce the notion of an f-reversal-bounded counter machine for a monotone function $f\colon \mathbb{N} \to \mathbb{N}$. In such a machine, every run of length n makes at most $f(n)$ reversals. Our first main result is a dichotomy theorem: We show that for every monotone function f, one of the following holds: Either (i) f grows so slowly that every f-reversal bounded counter machine is already k-reversal bounded for some constant k or (ii) f belongs to $\Omega(\log(n))$ and reachability in f-reversal bounded counter machines is undecidable. This shows that classical reversal bounding already captures the decidable cases of f-reversal bounding for any monotone function f. The key technical ingredient is an analysis of the growth of small solutions of iterated compositions of Presburger-definable constraints. In our second contribution, we investigate whether imposing f-reversal boundedness improves the complexity of the reachability problem in vector addition systems with states (VASS). Here, we obtain an analogous dichotomy: We show that either (i) f grows so slowly that every f-reversal-bounded VASS is already k-reversal-bounded for some constant k or (ii) f belongs to $\Omega(n)$ and the reachability problem for f-reversal-bounded VASS remains Ackermann-complete. This result is proven using run amalgamation in VASS.

Overall, our results imply that classical restriction of reversal boundedness is a robust one.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Counter machines, reversal-bounded, reachability, decidability, complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.42

Funding Alain Finkel was funded by the Institut Universitaire de France and the Agence Nationale de la Recherche, grant BraVAS (ANR-17-CE40-0028). Khushraj Madnani and Rupak Majumdar were supported in part by the Deutsche Forschungsgemeinschaft project 389792660 TRR 248–CPEC.

1 Introduction

The undecidability of the reachability problem in general multicounter machines is well-known [37]. Given this, there has been a rich landscape of restricted counter models, which have been studied with the aim of obtaining decidable reachability, while retaining as much expressiveness as possible. One of the most prominent restrictions studied is that of *reversal-bounded counter machines* [24]. As the name suggests, reversal-bounded counter machines bound the number of times a counter can change from an *incrementing phase* to a *decrementing phase*, or vice-versa, during a run. In an incrementing phase, the counter

© Alain Finkel, Shankara Narayanan Krishna, Khushraj Madnani, Rupak Majumdar, and Georg Zetzsche;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 42; pp. 42:1–42:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

is only incremented; likewise, in a decrementing phase, it is only decremented. As already shown by Ibarra in 1978 [24], for a constant bound, reachability is decidable and, further, reachability relations are semilinear. The latter observation allows decision procedures for Presburger arithmetic to be used for the algorithmic analysis of reversal-bounded systems.

Partly motivated by these available tools, reversal-bounded counter machines have been studied intensively. They have been studied in terms of deciding basic properties [19], model-checking various logics [3,14], expressiveness [26,29], deciding language-theoretic properties [2], regular abstractions [39], extensions that allow arbitrary reversals below a fixed counter value [15] or a free counter [13] or a pushdown [21] (see also [25] for a survey). This research has revealed a wide range of applications. For example, they have been used for model-checking recursive programs with numeric data types [21]. As another example, an equivalent variant of reversal-bounded counter machines is the model of Parikh automata [28], originally introduced to decide monadic second order logic with certain cardinality constraints [28], but subsequently studied as a computational model in itself [4–9, 17, 18].

Another decidable restriction of counter machines is that of vector addition systems with states (VASS). These allow arbitrary reversals, but have no zero tests. VASS are a standard model for analyzing concurrent systems. Unfortunately, the reachability problem in VASS is Ackermann-complete [11, 32]. However, imposing reversal-boundedness on VASS reduces the complexity down to NP: Reachability for reversal-bounded machines is NP-complete [19]. This has a number of implications. For example, one can efficiently analyze flat counter systems [36] by turning them into reversal-bounded systems. Flat counter systems, in turn, are the basis for flat acceleration techniques [1]. Furthermore, being (not flat but) flattable is equivalent to having semilinear reachability sets in VASS [31].

Thus, reversal bounding has turned out to be a fruitful restriction that ensures decidability in counter machines, and that reduces the computational complexity significantly in the case of VASS.

Infrequent reversals. Given this success of reversal-bounding counter machines, it seems natural to explore whether there are weaker conditions one can impose on reversals that still retain some decidability, say for the reachability problem. In this work, we explore the idea of requiring reversals to be *infrequent*, meaning the number of reversals is small *in relation to the length of the run*, as opposed to being a fixed constant independent of the length of the run. More specifically, for a monotone function $f\colon \mathbb{N} \to \mathbb{N}$, we call a counter machine *f-reversal-bounded* if in every run of length n, the machines makes at most $f(n)$ reversals. In other words, every run of length n decomposes into at most $f(n)$ phases, where each phase has a particular direction for each counter: Inside a phase, each counter can either (i) not be decremented or (ii) not be incremented. Clearly, this generalizes the classical notion of reversal-boundedness, which takes f to be a constant function. With this notion, we first investigate the following question:

For which monotone functions $f\colon \mathbb{N} \to \mathbb{N}$ do f-reversal-bounded counter machines have a decidable reachability problem?

Moreover, we study whether such a relaxed notion can serve to reduce the complexity of reachability in VASS:

For which monotone functions $f\colon \mathbb{N} \to \mathbb{N}$ do f-reversal-bounded VASS have a low-complexity reachability problem?

Contribution. For both questions above, we provide complete, but negative answers. First, we show that there are functions f that guarantee decidable reachability (e.g., $n \mapsto \log(\log(n))$), but they must grow so slowly that for every counter machine that is f-reversal-bounded, there exists a number $k \in \mathbb{N}$ such that the machine is already k-reversal-bounded. The main technical contribution used in proving this was the analysis of growth of minimal solution of k-fold composition of Presburger definable (i.e. semilinear) relations. On the other hand, if the function f grow fast enough (e.g., $n \mapsto \log(n)$), the reachability problem is undecidable.

An analogous situation holds for the complexity of VASS reachability: There are functions f for which f-reversal-bounded VASS have lower complexity than Ackermann (e.g., $n \mapsto \log(n)$), but these grow so slowly that for every VASS that is f-reversal-bounded, there exists some $k \in \mathbb{N}$ so that the VASS is k-reversal-bounded. On the other hand, if f grows sufficiently fast, the reachability problem is again Ackermann-complete.

In short, we show that there is no monotone function that properly relaxes the reversal-boundedness condition and still either (i) guarantees decidable reachability in counter machines or (ii) lowers the complexity of reachability in VASS.

Our results. Let us make our results precise. For every monotone function $f\colon \mathbb{N} \to \mathbb{N}$, we consider two decision problems. First, $\mathsf{Reach}(f)$ is the following:

> The problem $\mathsf{Reach}(f)$:
> **Given** An f-reversal-bounded counter machine CM and a configuration c.
> **Question** Can CM reach c from its initial configuration?

Second, the problem $\mathsf{Reach}_{\mathsf{VASS}}(f)$ restricts $\mathsf{Reach}(f)$ to f-reversal-bounded VASS:

> The problem $\mathsf{Reach}_{\mathsf{VASS}}(f)$:
> **Given** An f-reversal-bounded VASS $\mathcal{V}$ and a configuration c.
> **Question** Can $\mathcal{V}$ reach c from its initial configuration?

Now our two questions above become:

1. For which monotone functions $f\colon \mathbb{N} \to \mathbb{N}$ is $\mathsf{Reach}(f)$ decidable?
2. For which monotone functions $f\colon \mathbb{N} \to \mathbb{N}$ does $\mathsf{Reach}_{\mathsf{VASS}}(f)$ have lower complexity?

We say that a monotone function $f\colon \mathbb{N} \to \mathbb{N}$ is *essentially bounded (for counter machines)* if for every counter machine CM that is f-reversal-bounded, there exists a number $k \in \mathbb{N}$ such that CM is k-reversal-bounded. Our first main result is a dichotomy for monotone functions:

▶ **Theorem 1.1.** *Let $f\colon \mathbb{N} \to \mathbb{N}$ be a monotone function. Then exactly one of the following holds: Either (i) f is essentially bounded or (ii) f belongs to $\Omega(\log n)$, and $\mathsf{Reach}(f)$ is undecidable.*

To state our second main result, we say that f is *essentially bounded for VASS* if for every VASS $\mathcal{V}$ that is f-reversal-bounded, there exists a number $k \in \mathbb{N}$ such that $\mathcal{V}$ is k-reversal bounded. Our second main result is the following dichotomy:

▶ **Theorem 1.2.** *Let $f\colon \mathbb{N} \to \mathbb{N}$ be a monotone function. Then exactly one of the following holds: Either (i) f is essentially bounded for VASS or (ii) f belongs to $\Omega(n)$, and $\mathsf{Reach}(f)$ is Ackermann-complete.*

In other words, by relaxing the definition of "bounded reversal" to "infrequent reversal", we either get counter machines (or VASS) with undecidable (or Ackermann-hard) reachability problem or get a counter machines (or VASS) that are already reversal bounded. Hence, the class of machines within the purview of the classical reversal bounded restriction is already robust. That is, the notion of infrequent reversals does not give us any new decidable (resp., computationally easier) class of counter machines (resp., VASSes).

2 Preliminaries

Notations. We denote the set of all natural numbers (resp., integers) with $\mathbb{N}$ (resp., $\mathbb{Z}$). Given any set X, we write X^d to denote the set of all vectors of dimension d whose elements are in X. We write $\mathbf{0}^d$ for the vector all of whose entries are 0. We omit the superscript d from $\mathbf{0}^d$, when the dimension is clear from the context. Given $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{Z}^d$, we define $\boldsymbol{v}_1 \leq \boldsymbol{v}_2$ iff for all $i \in \{1, \ldots, d\}$, $\boldsymbol{v}_1[i] \leq \boldsymbol{v}_2[i]$. Moreover, $\boldsymbol{v}_1 + \boldsymbol{v}_2$ (resp., $\boldsymbol{v}_1 - \boldsymbol{v}_2$) is the vector $\boldsymbol{v}$ such that for all $i \in \{1, \ldots, d\}$, we have $\boldsymbol{v}[i] = \boldsymbol{v}_1[i] + \boldsymbol{v}_2[i]$ (resp., $\boldsymbol{v}[i] = \boldsymbol{v}_1[i] - \boldsymbol{v}_2[i]$). For a function $g\colon \mathbb{N} \to \mathbb{N}$, we write $\Omega(g)$ for the class of functions $f\colon \mathbb{N} \to \mathbb{N}$ for which there exist constants c, n_0 such that for all $n \geq n_0$, we have $f(n) \geq c \cdot g(n)$. As is customary, we write $\Omega(n)$ (resp. $\Omega(\log n)$) for the class $\Omega(g)$ with $g\colon n \mapsto n$ (resp. $g\colon n \mapsto \log(n)$).

Counter Machines and Vector Addition Systems with States. A *Counter Machine* is a 4-tuple $\mathsf{CM} = (d, Q, \Delta, q_0)$ where $d \in \mathbb{N}$ is the dimension, Q is a finite set of control states, and $\Delta \subseteq Q \times \mathsf{INST} \times Q$ is a finite set of transitions where $\mathsf{INST} = \mathbb{Z}^d \cup \{\mathsf{C}_j \stackrel{?}{=} 0 \mid j \in \{1, \ldots d\}\}$ is the set of all instructions, and $q_0 \in Q$ is the initial state.

An instruction is either some vector $\boldsymbol{z} \in \mathbb{Z}^d$, or a test of the form "$\mathsf{C}_k \stackrel{?}{=} 0$" where $k \in \{1, 2, \ldots d\}$. Instructions of the form $\mathsf{C}_k \stackrel{?}{=} 0$ are called zero tests.

A *configuration* of CM is a pair $(q, \boldsymbol{v})$ where $q \in Q$ and $\boldsymbol{v} \in \mathbb{N}^d$. The *initial configuration* is $(q_0, \mathbf{0})$. Any transition $t = (q, inst, q')$ induces a successor (partial) function $\mathsf{Succ}_t : Q \times \mathbb{N}^d \to Q \times \mathbb{N}^d$ defined as follows. If $\boldsymbol{z} \in \mathbb{Z}^d$ then $\mathsf{Succ}_t((q, \boldsymbol{v})) = (q', \boldsymbol{v}')$ where $\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{z}$. If $\boldsymbol{z}$ is a zero test instruction of the form "$\mathsf{C}_k \stackrel{?}{=} 0$" then $\mathsf{Succ}_t((q, \boldsymbol{v})) = (q', \boldsymbol{v})$ if $\boldsymbol{v}[k] = 0$ and is not defined otherwise. This successor function can be lifted to Δ to get a step relation $\to_{\mathsf{CM}}$, such that, for any pair of configurations $\mathcal{C}, \mathcal{C}'$, we have $\mathcal{C} \to_{\mathsf{CM}} \mathcal{C}'$ iff there exists a transition $t \in \Delta$ such that $\mathsf{Succ}_t(\mathcal{C}) = \mathcal{C}'$. We sometimes use the term counters and coordinates interchangeably in the context of vectors.

An (initialized) *run* of CM is a sequence of configurations $(q_0, \boldsymbol{v}_0)(q_1, \boldsymbol{v}_1) \ldots (q_n, \boldsymbol{v}_n)$ such that $\boldsymbol{v}_0 = \mathbf{0}$ and for every $0 < j \leq n$, $(q_{j-1}, \boldsymbol{v}_{j-1}) \to_{\mathsf{CM}} (q_j, \boldsymbol{v}_j)$ holds. If there exists such a run we say that $(q_n, \boldsymbol{v}_n)$ is *reachable* from $(q_0, \boldsymbol{v}_0)$ and denote it as $(q_0, \boldsymbol{v}_0) \stackrel{*}{\to}_{\mathsf{CM}} (q_n, \boldsymbol{v}_n)$. We drop the subscript from $\to_{\mathsf{CM}}$ and $\stackrel{*}{\to}_{\mathsf{CM}}$ when the counter machine is clear from context.

Vector Addition System with States (*VASS*) is a subclass of counter machines where the transitions are restricted to not have zero tests.

The *reachability problem* for counter machines asks:
Given Given a configuration $(q, \boldsymbol{w})$ of a counter machine CM.
Question Does $(q_0, \mathbf{0}) \stackrel{*}{\to}_{\mathsf{CM}} (q, \boldsymbol{w})$ hold?

Reversal Boundedness. Informally, a counter is said to be in an increasing phase in a part of a run, if the value of that counter does not decrease in any step of the given part. Likewise the counter is in a decreasing phase in a part of a run, if its value does not increase in any step of the given part. A reversal is a step in a given run, where one or more counters switch from increasing to decreasing phase, or vice versa. Moreover, for $r \in \mathbb{N}$, a counter machine CM is r-reversal bounded if for any run of CM, all of its runs have at most r reversals.

Let us make this more formal. Let $\mathsf{CM} = (d, Q, \Delta)$ be any counter machine. A *mode vector* is a vector $\boldsymbol{m} \in \mathbb{Z}^d$ where every entry is -1 or 1. A mode vector describes the direction in which a counter can change: If $\boldsymbol{m}(i) = 1$, then this means counter i cannot be decremented. Similarly, $\boldsymbol{m}(i) = -1$ means it cannot be incremented. A step $(q, \boldsymbol{v}) \to (q', \boldsymbol{w})$ that adheres to this is called *consistent* with $\boldsymbol{m}$. A run is *consistent* with $\boldsymbol{m}$ if all its steps in the run are consistent with $\boldsymbol{m}$.

To define phases, consider a run $\rho = (q_0, \boldsymbol{v}_0)(q_1, \boldsymbol{v}_1) \dots (q_n, \boldsymbol{v}_n)$ of CM. We say that a segment $\rho' = (q_i, \boldsymbol{v}_i)(q_{i+1}, \boldsymbol{v}_{i+1}) \dots (q_j, \boldsymbol{v}_j)$ of ρ is a *phase* of ρ, iff, ρ' is consistent with some mode vector $\boldsymbol{m}$. Intuitively, a phase is a part of the run where each counter $1 \leq i \leq d$ is either increasing in all steps, or is decreasing in all steps. ρ' is said to be the *maximal* phase of ρ iff it no longer remains a phase on extending it. That is, for $\rho' = (q_i, \boldsymbol{v}_i)(q_{i+1}, \boldsymbol{v}_{i+1}) \dots (q_j, \boldsymbol{v}_j)$ and $j < n$, $\rho'.(q_{j+1}, \boldsymbol{v}_{j+1})$ is no longer a phase; likewise, for $\rho' = (q_i, \boldsymbol{v}_i)(q_{i+1}, \boldsymbol{v}_{i+1}) \dots (q_j, \boldsymbol{v}_j)$ and $i > 1$, $(q_{i-1}, \boldsymbol{v}_{i-1}).\rho'$ is also no more a phase of ρ.

A run ρ contains r *reversals* iff ρ can be decomposed into r maximal phases. That is, $\rho = \rho_1.\rho_2.\dots.\rho_r$ such that for each $i \in \{1, \dots r\}$, ρ_i is a maximal phase of ρ.

A counter machine CM is said to be *r-reversal bounded* iff all its runs contain at most r reversals.

3 Counter machines: Decidable case

Our proof of Theorem 1.1 consists of showing two propositions:

▶ **Proposition 3.1.** *If there exists an f-reversal-bounded counter machine that is not reversal-bounded, then f belongs to $\Omega(\log(n))$.*

▶ **Proposition 3.2.** *If f belongs to $\Omega(\log(n))$, then $\mathsf{Reach}(f)$ is undecidable.*

Together, these clearly imply Theorem 1.1: Proposition 3.2 implies that if f belongs to $\Omega(\log(n))$, then $\mathsf{Reach}(f)$ is undecidable. Moreover, Proposition 3.1 tells us that if f does not belong to $\Omega(\log(n))$, then any f-reversal-bounded counter machine must already be reversal-bounded. In this section, we prove Proposition 3.1. In Section 4, we will then prove Proposition 3.2.

We start with defining semilinear sets and Presburger arithmetic.

Semilinear Sets. A set $S \subseteq \mathbb{N}^d$ is called *linear* iff there exists finitely many vectors $\boldsymbol{v}_0, \boldsymbol{v}_1, \dots \boldsymbol{v}_k \in \mathbb{N}^d$ such that

$$S = \{\boldsymbol{v}_0 + \sum_{i=1}^{k} c_i \cdot \boldsymbol{v}_i | c_1, c_2, \dots c_n \in \mathbb{N}\}$$

A *semilinear* set is a finite union of linear sets. Linear and semilinear relations are defined similarly.

Presburger Arithmetic. Presburger Arithmetic is defined to be the first-order theory of natural numbers endowed with the addition operation $(+)$, comparison $(<)$, and equality $(=)$ predicates. All the predicates have the usual meaning over the natural numbers. For the sake of simplicity, we allow constants in $\mathbb{N}$, as well as the multiplication of a constant by a variable as terms. Note that these terms are expressible using basic Presburger formulae. The following theorems connect Presburger constraints, semilinearity and reversal boundedness.

▶ **Theorem 3.3** (Ginsburg & Spanier [16]). *A relation is definable in Presburger arithmetic if and only if it is semilinear.*

▶ **Theorem 3.4** (Ibarra [24]). *Given any k-reversal bounded counter machine* CM *(for some k), its reachability relation is semilinear.*

Iterations of semilinear sets. Proposition 3.1 says that if a counter machine can make an unbounded number of reversals, but is f-reversal-bounded, then for every n, f must allow for a run of length n with at least $\Omega(\log(n))$ reversals. To this end, we will show that for any k, there exists a run with k reversals and of length at most c^k, for some constant c.

Using standard techniques for reversal-bounded counter machines, it is not difficult to show that there exists such a run of length $2^{k^{O(1)}}$: It is easy to construct an existential Presburger formula Φ_k of size $O(k)$ such that $\Phi_k(n)$ represents a run of length n. Then it follows (for example, from [20, Theorem 2]) that there exists a run of length at most $2^{k^{O(1)}}$. However, in order to prove our complete dichotomy, we need to refine these techniques to prove an upper bound of c^k for a constant c.

For a relation $R \subseteq \mathbb{N}^d \times \mathbb{N}^d$, we define R^k to be the k-fold composition of R, meaning $R^1 := R$ and $R^{k+1} := R^k \circ R$ for $k \geq 1$. For a vector $\boldsymbol{z} \in \mathbb{Z}^d$, $\boldsymbol{z} = (z_1, \ldots, z_d)$, we define its norm as $\|\boldsymbol{z}\| = \max\{|z_i| \mid i \in \{1, \ldots, d\}$. The key step in getting a c^k upper bound is the following lemma.

▶ **Lemma 3.5.** *For any semilinear $R \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$ and $S \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$, there is a constant $c \in \mathbb{N}$ such that for every $k \in \mathbb{N}$, if $R^k \cap S \neq \emptyset$, then there is a $\boldsymbol{z} \in R^k \cap S$ with $\|\boldsymbol{z}\| \leq c^k$.*

In the proof of Lemma 3.5, we will rely on the following bound on solution sizes of systems of linear inequalities due to [38, Corollary 1]. Here, for a matrix $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$, with entries a_{ij} ($i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$), we follow [38] in defining the norm

$$\|\boldsymbol{A}\|_{1,\infty} = \max_i \left\{ \sum_{j=1}^{n} |a_{ij}| \right\}.$$

▶ **Theorem 3.6** (Pottier 1991). *Let $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$ and $\boldsymbol{a} \in \mathbb{Z}^m$. If there exists an $\boldsymbol{x}$ with $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{a}$, then there exists one with $\|\boldsymbol{x}\| \leq (2 + \|\boldsymbol{A}\|_{1,\infty} + \|\boldsymbol{a}\|)^m$.*

Proof of Lemma 3.5. We begin by expressing R and S using a system of linear Diophantine inequalities. Since they are semilinear, they can be expressed using existential Presburger formulae $\exists \boldsymbol{u} \colon \varphi_R(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y})$ and $\exists \boldsymbol{u} \colon \varphi_S(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of d variables each, and $\boldsymbol{u}$ is a vector of existentially quantified variables. Note that the quantified variables $\boldsymbol{u}$ are in different positions in φ_R and φ_S; this will be convenient later when constructing matrices. Now φ_R and φ_S are Boolean combinations of linear Diophantine inequalities. By bringing these into DNF, we can write

$$\varphi_R(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}) \iff \bigvee_{i=1}^{r} \boldsymbol{A}_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}) \leq \boldsymbol{a}_i$$

$$\varphi_S(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}) \iff \bigvee_{i=1}^{s} \boldsymbol{B}_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}) \leq \boldsymbol{b}_i$$

for some matrices $\boldsymbol{A}_i, \boldsymbol{B}_i \in \mathbb{Z}^{\ell \times m}$ and $\boldsymbol{a}_i, \boldsymbol{b}_i \in \mathbb{Z}^{\ell}$. Here m is the combined number of variables across all of $\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{y}$ and ℓ is the (maximal) number of inequalities needed for φ_R and φ_S.

$$\begin{pmatrix} \begin{bmatrix} & \bar{\boldsymbol{A}}_1 & \end{bmatrix} & & & & \\ & \begin{bmatrix} & \bar{\boldsymbol{A}}_2 & \end{bmatrix} & & & \\ & & \ddots & & \\ & & & \begin{bmatrix} & \bar{\boldsymbol{A}}_k & \end{bmatrix} & \\ \begin{bmatrix} \boldsymbol{C} & \end{bmatrix} & & & \begin{bmatrix} & \boldsymbol{D} & \end{bmatrix} & \end{pmatrix} \cdot \boldsymbol{w} \leq \begin{pmatrix} \begin{bmatrix} \bar{\boldsymbol{a}}_1 \end{bmatrix} \\ \begin{bmatrix} \bar{\boldsymbol{a}}_2 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \bar{\boldsymbol{a}}_k \end{bmatrix} \\ \begin{bmatrix} \boldsymbol{b} \end{bmatrix} \end{pmatrix}$$

Figure 1 Inequality in proof of Lemma 3.5. All other elements not in matrices $\boldsymbol{A_1}, \dots \boldsymbol{A_k}, \boldsymbol{C}, \boldsymbol{D}$ are 0.

Suppose there is a vector $\boldsymbol{z} \in R^k \cap S$. Then a system of linear inequalities of the form sketched in Figure 1 has a solution. Here, for each $i \in \{1, \dots, k\}$, there is some $j \in \{1, \dots, r\}$ such that $\bar{\boldsymbol{A}}_i = \boldsymbol{A}_j$ and $\bar{\boldsymbol{a}}_i = \boldsymbol{a}_j$. Moreover, the overlap between $\bar{\boldsymbol{A}}_i$ and $\bar{\boldsymbol{A}}_{i+1}$ is exactly the right-most d columns of $\bar{\boldsymbol{A}}_i$ and the left-most d columns of $\bar{\boldsymbol{A}}_{i+1}$. Moreover, there is some $j \in \{1, \dots, s\}$ such that the matrix $\boldsymbol{C}$ consists of the first d columns of $\boldsymbol{B}_j$ and $\boldsymbol{D}$ consists of the last $m - d$ columns of $\boldsymbol{B}_j$ (and the overlap to $\bar{\boldsymbol{A}}_k$ is exactly in the right-most d columns of $\bar{\boldsymbol{A}}_k$).

Let $\boldsymbol{A}$ be the matrix in Figure 1 and $\boldsymbol{a}$ be the vector on the right-hand side of Figure 1. Observe that $\boldsymbol{A}$ has $(k+1) \cdot \ell$ rows. By its choice, the inequality $\boldsymbol{A}\boldsymbol{w} \leq \boldsymbol{a}$ has a solution. Thus, by Theorem 3.6, we have a solution $\boldsymbol{w}'$ with $\|\boldsymbol{w}'\| \leq (2 + \|\boldsymbol{A}\|_{1,\infty} + \|\boldsymbol{a}\|)^{(k+1)\cdot\ell}$. However, since each row sum of $\boldsymbol{A}$ is a row sum of either (i) a matrix $\boldsymbol{A}_j$ for some $j \in \{1, \dots, r\}$ or (ii) a matrix $\boldsymbol{B}_j$ for some $j \in \{1, \dots, s\}$, we know that

$$\|\boldsymbol{A}\|_{1,\infty} \leq \max\left(\{\|\boldsymbol{A}_j\|_{1,\infty} \mid j \in \{1, \dots, r\}\} \cup \{\|\boldsymbol{B}_j\|_{1,\infty} \mid j \in \{1, \dots, s\}\}\right) \tag{1}$$

and moreover

$$\|\boldsymbol{a}\| \leq \max\left(\{\|\boldsymbol{a}_j\| \mid j \in \{1, \dots, r\}\} \cup \{\boldsymbol{b}_j \mid j \in \{1, \dots, s\}\}\right). \tag{2}$$

We pick M to be an upper bound of the right-hand sides of (1) and (2). Then we have

$$\|\boldsymbol{w}'\| \leq (2 + \|\boldsymbol{A}\|_{1,\infty} + \|\boldsymbol{a}\|)^{(k+1)\ell} \leq (2 + 2M)^{(k+1)\ell} \leq (2 + 2M)^{2k\ell}.$$

Thus, setting $c := (2 + 2M)^{2\ell}$ gives us the desired bound: By projecting $\boldsymbol{w}'$ to appropriate components, we obtain a vector $\boldsymbol{z}' \in R^k \cap S$ with $\|\boldsymbol{z}'\| \leq \|\boldsymbol{w}'\| \leq c^k$. ◀

Reachability relations. Our next step is to apply the well-known fact that the reachability relation of runs along a single phase of a counter machine is Presburger-definable. Recall that a *phase* is a run in which no counter reverses. More precisely, our next lemma makes a slightly different (but equally simple) claim: The reachability relation along runs that consist of a single phase followed by a single transition (that leaves that phase) is Presburger-definable.

Given a counter machine CM with d counters, we define the relation $R_{\mathsf{CM}} \subseteq \mathbb{N}^{1+2d} \times \mathbb{N}^{1+2d}$ as follows. We have $(i, \boldsymbol{m}, \boldsymbol{x}, j, \boldsymbol{m'}, \boldsymbol{y}) \in R_{\mathsf{CM}}$ if and only if there exists a run ρ consisting of transitions $t_1 \dots t_{n+1}$ such that there exists a $\boldsymbol{x} \in \mathbb{N}^d$ and a state $q_k \in Q$ such that

1. the run $t_1 \cdots t_n$ is a run from $(q_i, \boldsymbol{x})$ to $(q_k, \boldsymbol{x'})$ that is consistent with $\boldsymbol{m}$ and
2. t_{n+1} is not consistent with $\boldsymbol{m}$, but t_{n+1} is consistent with $\boldsymbol{m'}$
3. t_{n+1} leads from $(q_k, \boldsymbol{x'})$ to $(q_j, \boldsymbol{y})$.

Thus, R_{CM} is the reachability relation for a run consistent with $\boldsymbol{m}$, plus one transition that is consistent with $\boldsymbol{m'}$, and such that the last step reverses some counter. Moreover, we also encode the mode vector in the components of R_{CM}. The following is entirely standard, but we include a proof for completeness.

▶ **Lemma 3.7.** *For every counter machine* CM*, the relation* R_{CM} *is Presburger-definable.*

Proof. It suffices to prove that for any mode vectors $\boldsymbol{m}$ and $\boldsymbol{m'}$ in $\mathbb{N}^d$ and any $i, j \in \mathbb{N}$, the set $R' \subseteq \mathbb{N}^d \times \mathbb{N}^d$ of all $(\boldsymbol{x}, \boldsymbol{y})$ with $(i, \boldsymbol{m}, \boldsymbol{x}, j, \boldsymbol{m'}, \boldsymbol{y}) \in R_{\mathsf{CM}}$ is Presburger-definable. This is because there are only finitely many choices for $i, j, \boldsymbol{m}, \boldsymbol{m'}$. However, for given $i, j, \boldsymbol{m}, \boldsymbol{m'}$, it is easy to construct a reversal-bounded counter machine CM' with d counters and states s, t such that in CM', we have $s(\boldsymbol{x})$ can reach $t(\boldsymbol{y})$ if and only if $(\boldsymbol{x}, \boldsymbol{y})$ is Presburger-definable. To this end, CM' simulates transitions consistent with the mode $\boldsymbol{m}$ in CM, and then it simulates one more transition consistent with mode $\boldsymbol{m'}$ (which is not consistent with $\boldsymbol{m}$). Since the reachability relation in every reversal-bounded counter machine is Presburger-definable [24], R' is Presburger-definable and the result follows. ◀

With Lemmas 3.5 and 3.7 in hand, we are now ready to prove Proposition 3.1.

Proof of Proposition 3.1. Suppose CM is an f-reversal-bounded counter machine that is not reversal bounded. Without loss of generality, we assume that CM has one counter that is incremented in every step; this counter thus always holds the length of the run.

We consider the following function: $L \colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$, where for each k, $L(k)$ is the length of the shortest run that contains (at least) k reversals. If there is no such run, we set $L(k) := \infty$. Since CM is not reversal-bounded, we have $L(k) \in \mathbb{N}$ for every $k \in \mathbb{N}$.

Note that any run of CM starts in the mode $\mathbf{1}^d$, because initially, no counter can be decremented. Moreover, all counters are zero initially, and we start in q_0. This motivates the following definition. Let $S \subseteq \mathbb{Z}^{1+2d} \times \mathbb{Z}^{1+2d}$ be the set $S = \{(0, \mathbf{1}^d, \mathbf{0}^d)\} \times \mathbb{Z}^{1+2d}$ of vectors. In other words, S contains those vectors where the first entry is 0, the next d entries contain 1, and the next d entries contain 0 (and the last $1 + 2d$ entries are unrestricted). Then clearly, a configuration $(q_j, \boldsymbol{y})$ can be reached using exactly k reversals if and only if

$$(0, \mathbf{1}^d, \mathbf{0}^d, j, \boldsymbol{m'}, \boldsymbol{y}) \in R^k_{\mathsf{CM}}$$

for some mode vector $\boldsymbol{m'}$. Thus, by our assumption that CM is not reversal-bounded, we have $R^k_{\mathsf{CM}} \cap S \neq \emptyset$ for every $k \in \mathbb{N}$. Now let c be the constant provided by Lemma 3.5 for R_{CM} and S. Then for every $k \in \mathbb{N}$, there is a vector $\boldsymbol{z}_k \in R^k_{\mathsf{CM}} \cap S$ with $\|\boldsymbol{z}_k\| \leq c^k$. Since the length of each run is encoded in a counter, the length of the corresponding run is thus $\leq \|\boldsymbol{z}_k\| \leq c^k$. In particular:

$$L(k) \leq c^k \ \forall k \in \mathbb{N}$$

Now observe that for every $k \in \mathbb{N}$, we have $f(L(k)) \geq k$, because CM is f-reversal-bounded and there exists a run of length $L(k)$ with at least k reversals. Since f is monotone and $L(k) \leq c^k$, we have $k \leq f(L(k)) \leq f(c^k)$ for every $k \in \mathbb{N}$. Again by monotonicity of f, this implies that for every n, we have $f(n) \geq \log(n)/\log(c)$. Thus f belongs to $\Omega(\log(n))$. ◀

4 Counter machines: Undecidable case

In this section, we prove Proposition 3.2. That is, given any monotone function $f : \mathbb{N} \to \mathbb{N}$ such that f belongs to $\Omega(\log(n))$, we show that $\mathsf{Reach}(f)$ is undecidable. To this end, we show that any counter machine can be simulated by an f-reversal bounded counter machine.

Making f concrete. Instead of working with an arbitrary function f in $\Omega(\log(n))$, it will be convenient to work with a concrete function of the form $c \cdot \log$. Let us now see why it suffices to show undecidability of $\mathsf{Reach}(c \cdot \log)$. The fact that f belongs to $\Omega(\log n)$ means that there exist $c, k \in \mathbb{N}$ such that $f(n) \geq c \cdot \log(n)$ for all $n \geq k$. This implies that $\mathsf{Reach}(c \cdot \log)$ reduces to $\mathsf{Reach}(f)$: Given a $c \cdot \log$-reversal-bounded machine, we modify it to begin with k fresh steps that do not reverse any counter. Then clearly, the new machine is f-reversal-bounded. Hence, to show $\mathsf{Reach}(f)$ is undecidable it suffices to show that $\mathsf{Reach}(c \cdot \log)$ is undecidable.

Making the constant concrete. It will be even more convenient to show undecidability in the case $c = 11$. Our next lemma argues that this suffices.

▶ **Lemma 4.1.** *For every monotone function $g \colon \mathbb{N} \to \mathbb{N}$ and every constant $c \in \mathbb{N}$, $\mathsf{Reach}(g)$ is decidable if and only if $\mathsf{Reach}(c \cdot g)$ is decidable*

Proof. Since every g-reversal-bounded counter machine is also a $c \cdot g$-reversal-bounded machine, $\mathsf{Reach}(g)$ trivially reduces to $\mathsf{Reach}(c \cdot g)$. For the converse, we will prove that $\mathsf{Reach}(c \cdot g)$ reduces to $\mathsf{Reach}(g)$. Given any $c \cdot g$-reversal bounded counter machine $\mathsf{CM} = (d, Q, \Delta, q_0)$ construct $\mathsf{CM}_c = (d, Q_c, \Delta_c, (q_0, 0))$ such that it adds c dummy transitions between any two consecutive steps of CM. Formally, $Q_c = Q \times \{0, \ldots c\}$, and Δ_c is the smallest set containing transitions, $((q, i), \mathbf{0}^d, (q, i+1))$ (where $i \in \{0, \ldots, c-1\}$), and $((q, c), inst, (q', 0))$ where $(q, inst, q') \in \Delta$. Then, $c \cdot g$-reversal-boundedness of the input machine clearly implies g-reversal-boundedness of the resulting machine. Moreover, $(q, \boldsymbol{w})$ is reachable from $(q_0, \mathbf{0})$ in CM if and only if $((q, 0), \boldsymbol{w})$ is reachable from $((q_0, 0), \mathbf{0})$. ◀

Now Lemma 4.1 indeed implies that, decidability of $\mathsf{Reach}(11 \cdot \log)$ is equivalent to decidability of $\mathsf{Reach}(c \cdot \log)$. Hence, we just need to show that $\mathsf{Reach}(11 \cdot \log)$ is undecidable.

Proving $\mathsf{Reach}(11 \cdot \log)$ is undecidable – Main Step. In this step, we show that any counter machine $\mathsf{CM} = (d, Q, \Delta, q_0)$ can be simulated by an $11 \cdot \log$-reversal bounded counter machine $\mathsf{CM}' = (d', Q', \Delta', q_0')$. Like Lemma 4.1, we will add dummy steps between any two steps simulating CM. But unlike Lemma 4.1, we need to add more and more dummy steps after simulating every step of CM. More precisely, between the simulation of the Nth and $(N+1)^{th}$ step of CM, we need to add exponentially many (in N) dummy steps, using reversals which are polynomial in N. To do this, we construct a gadget for every state $q \in Q$, which does the above, before simulating any outgoing transition from q using 4 additional auxiliary counters. We now give the formal construction followed by the proof of correctness.

Formal construction of CM'. Given $\mathsf{CM} = (d, Q, \Delta, q_0)$, we construct a counter machine $\mathsf{CM}' = (Q', d', \Delta', (q_0, 0))$ where $Q' = Q \times \{0, \ldots 10\}$, $d' = d + 4$, and Δ' is defined a little later. For the sake of simplicity, we assume $d = 2$. Hence, counters $\mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5, \mathsf{C}_6$ are the auxiliary counters used to add the required dummy steps.

Transitions simulating steps of CM. For any state $q \in Q$, transitions exiting q in CM are simulated by transitions exiting $(q, 10) \in Q'$. More precisely, $(q, [a_1, a_2], q') \in \Delta$ iff $((q, 10), [a_1, a_2, 0, 0, 0], (q', 0)) \in \Delta'$.

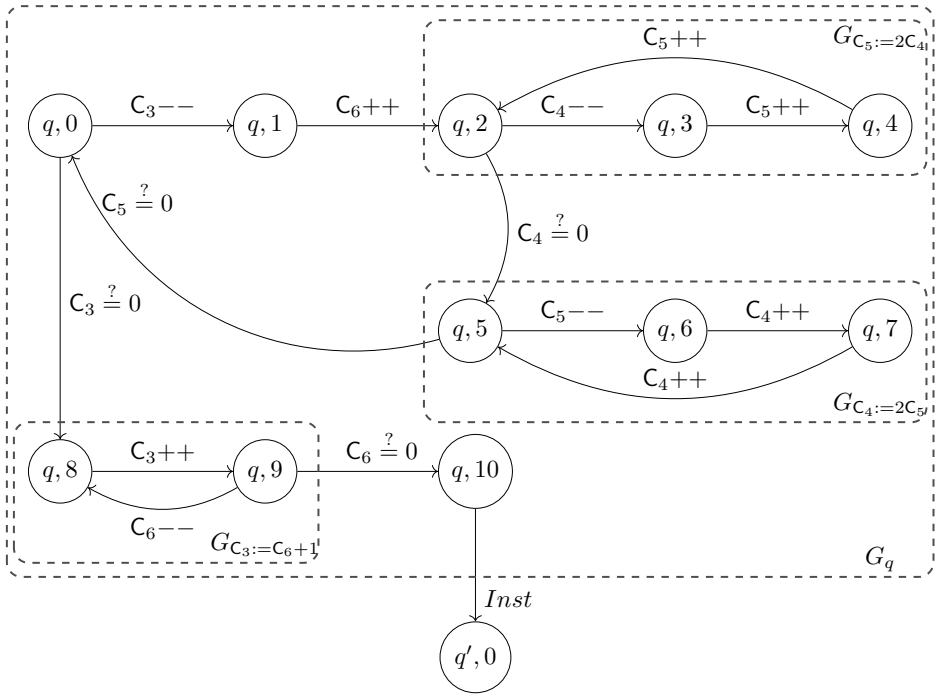


Figure 2 Diagram showing simulation of transition $q \xrightarrow{Inst} q'$. G_q is the gadget for state q. For the sake of readability, for all $1 \leq i \leq 6$, we write $\mathsf{C}_i - -$ ($\mathsf{C}_i + +$) for vector $\boldsymbol{z}$ such that for all $1 \leq j \leq i$, $\boldsymbol{z}[j] = -1$ ($\boldsymbol{z}[j] = +1$) if $j = i$ and $\boldsymbol{z}[j] = 0$, otherwise.

Transitions adding dummy steps. The rest of the transitions are those appearing within the gadgets. For every state $q \in Q$, we construct a gadget G_q, as shown in Figure 2, such that before imitating any outgoing transition from state q, G_q induces the required number of dummy steps.

Enforcing long runs with less reversals, G-Invariant Property. We say that our gadget satisfies the *G-Invariant* property iff on entering the gadget with counter values $[a, b, s, t, 0, 0]$ we exit the gadget with the values of these counters as $[a, b, s + 1, 4^s \cdot t, 0, 0]$. Moreover, the number of reversals made within this gadget (including the transition outgoing from the gadget) is at most $4s + 5$.

Observe the gadget in Figure 2. The gadget adds dummy transitions before it simulates any transition of CM outgoing from p as follows: It performs an identical transition from $(q, 10)$, namely, $(q, 10) \xrightarrow{Inst} (q', 0)$. The transitions from $(q, 0)$ to $(q, 10)$ enforces the runs to be long enough to make sure that CM' is a $c \cdot$ log-reversal bounded counter machine. The analysis of CM' being 11 log-reversal bounded is shown a little later. More specifically, the run from $(q, 0)$ to $(q, 10)$ satisfies the G-Invariant property. Let the initial values of these auxiliary counters be $\mathsf{C}_3 = s, \mathsf{C}_4 = t, \mathsf{C}_5 = 0, \mathsf{C}_6 = 0$. Notice the following:

- When the control enters $(q, 2)$ it remains in the part of the gadget $G_{\mathsf{C}_5:=2\mathsf{C}_4}$ and exits this part by entering into $(q, 5)$ with the new value of C_4 as 0 and C_5 incremented by double of what the value of C_4 was when the control first entered the $G_{\mathsf{C}_5:=2\mathsf{C}_4}$ part. By symmetry, part $G_{\mathsf{C}_4:=2\mathsf{C}_5}$ increments C_4 by the double of the old value of C_5.
- For every decrement of C_3, C_6 is incremented, and the control enters $G_{\mathsf{C}_5:=2\mathsf{C}_4}$ followed by entering $G_{\mathsf{C}_4:=2\mathsf{C}_5}$. After this the control again comes back to $(q, 0)$.
- Hence, for each decrement of C_3, $\mathsf{C}_4 := 4\mathsf{C}_4$, C_6 is incremented and $\mathsf{C}_5 := 0$.

- The above continues till $\mathsf{C}_3 = 0$ (i.e. s times) after which $\mathsf{C}_6 = s$, $\mathsf{C}_3 = 0$, $\mathsf{C}_5 = 0$, $\mathsf{C}_4 = 4^s \cdot t$ and the control enters $(q, 8)$.
- From $(q, 8)$, for every increment of C_3, C_6 is decremented, and this continues until $\mathsf{C}_6 = 0$ at $(q, 9)$ after which $\mathsf{C}_3 = s + 1$ and the control enters $(q, 10)$. This is followed by simulating instructions from q as in the original machine CM.
- Hence the values of the auxiliary counters at $(q, 10)$ will be $\mathsf{C}_3 = s+1, \mathsf{C}_4 = 4^s \cdot t, \mathsf{C}_5 = 0, \mathsf{C}_6 = 0$, implying the satisfaction of the G-Invariant property for the given gadget.
- Finally notice the number of reversals made within this gadget. There are 2 reversals between entering and exiting the part $G_{\mathsf{C}_5:=2\mathsf{C}_4}$ of the gadget (the very first time C_5 is incremented and the very first time C_4 is decremented). Similarly, there are 2 reversals between entering and exiting the part $G_{\mathsf{C}_4:=2\mathsf{C}_5}$ and $G_{\mathsf{C}_3:=\mathsf{C}_6+1}$. The total number of times the control passes through the parts $G_{\mathsf{C}_5:=2\mathsf{C}_4}$ and $G_{\mathsf{C}_5:=2\mathsf{C}_4}$ is s each. Similarly, the control passes exactly once from the part $G_{\mathsf{C}_3:=\mathsf{C}_6+1}$ causing $4s + 2$ reversals. Moreover, there is exactly one reversal when transitions $(q, 0) \to (q, 1)$ and $(q, 1) \to (q, 2)$ are taken the very first time within the gadget. Finally, there can be at most one reversal while simulating $Inst$. Hence, there are at the most $4s + 5$ reversals.

This technique of moving tokens between different auxiliary counters and doubling was also used in [23] to show that VASS with 3 or more dimensions can have non-semilinear reachable sets. As they were interested in reach sets, they did not require all the runs to be large. On the other hand, we need long runs to show undecidability and with infrequent reversals. Hence, unlike [23], we need zero tests.

It is interesting to note that a VASS gadget satisfying the G-Invariant property cannot exist. If it did, it would imply that there is a $\log(n)$-reversal bounded VASS that does not have a constant number of reversals, which contradicts our Theorem 1.2.

Proving CM' is $11 \cdot \log$-reversal bounded. Assume that we start with $[a, b, 1, 1, 0, 0]$ as our initial configuration. Consider any run $\rho = (q_0, \boldsymbol{v}_0) \dots (q_n, \boldsymbol{v}_n)$ of CM', where $\boldsymbol{v}_0 = \mathbf{0}$.

Lower bounding the length of the run. Suppose ρ enters the gadget during the $(N+1)^{th}$ time (i.e., it has simulated N instructions of the CM and is about to simulate the $(N + 1)^{th}$ instruction). Let $(q_0, \boldsymbol{v}_0) \dots (q_\ell, \boldsymbol{v}_\ell)$ be the prefix of ρ such that, at the ℓth step, the Nth instruction of the CM was simulated. As our gadget preserves the G-Invariant property, we can inductively show that the value of C_4 in $\boldsymbol{v}_\ell$ i.e. $\boldsymbol{v}_\ell[4]$ is

$$t_N = 4 \cdot 4^2 \cdot 4^3 \cdot \dots \cdot 4^N = 2^{N^2+N}.$$

Since in the initial configuration, C_4 equals 1 and in each step of CM', C_4 is increased by at most 1, this means $\ell_N \geq t_N$.[1]

Upper bounding the number of reversals. The number of reversals k in ρ is at most the maximum possible number of reversals k_{N+1} at the time when the control exits the $N + 1^{th}$ gadget. That is,

$$k_{N+1} = \sum_{i=1}^{N+1} (4i + 5) = 4 \cdot \frac{(N + 1)(N + 2)}{2} + 5(N + 1)$$
$$= 2N^2 + 6N + 4 + 5N + 5 \leq 11N^2 + 11N \quad (3)$$

for any $N \geq 1$.[2] See Figure 3 for intuition.

[1] Note that the value of ℓ_N is much larger than t_N for higher values of N. In fact the number of dummy steps added within the Nth entry of the gadget itself is at least t_N. This is because C_4 becomes 0 at least once, within the gadget. Hence, to again reach the value t_N, it needs to execute at least t_N steps. Hence, the number of steps performed within the gadget itself is at least t_N.

[2] For $N \leq 1$, any counter machine will have zero reversals. Hence, the value of the function bounding the frequency of the reversal is important only for $N \geq 1$

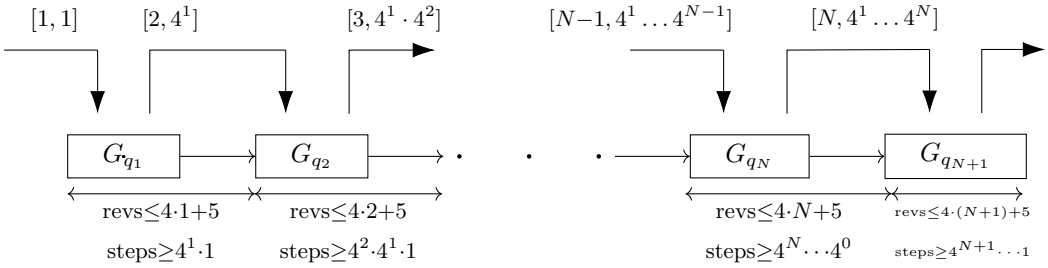


Figure 3 Figure showing the evaluation of counter values of auxiliary counters C_3 and C_4, which are used to estimate a lower bound on the length of the runs. The lower bounds on the steps, and the upper bound on the number of reversals performed within a gadget is mentioned below the corresponding gadget.

Hence, $k \leq k_{N+1} \leq 11 \cdot \log(t_N) \leq 11 \cdot \log(\ell) \leq 11 \cdot \log(n)$. Hence, CM' is $11 \cdot \log$-reversal bounded.

We have thus shown that $\mathsf{Reach}(11 \cdot \log)$ is undecidable and therefore, by Lemma 4.1, $\mathsf{Reach}(\log)$ is undecidable.

5 Vector addition systems

In this section, we prove Theorem 1.2. Let $\mathcal{V} = (d, Q, \Delta, q_0)$ be a d-dimensional vector addition system. A *step* is a triple $((p, \boldsymbol{u}), \boldsymbol{a}, (q, \boldsymbol{v})) \in (Q \times \mathbb{N}^d) \times \Delta \times (Q \times \mathbb{N}^d)$ with $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{a}$. A *run* is a sequence

$$((q_0, \boldsymbol{u}_1), \boldsymbol{a}_1, (q_1, \boldsymbol{v}_1))((q_1, \boldsymbol{u}_2), \boldsymbol{a}_2, (q_2, \boldsymbol{v}_2)) \cdots ((q_{n-1}, \boldsymbol{u}_n), \boldsymbol{a}_n, (q_n, \boldsymbol{v}_n))$$

of steps such that $\boldsymbol{v}_i = \boldsymbol{u}_{i+1}$ for each $1 \leq i < n$ and $\boldsymbol{u}_1 = \boldsymbol{0}$.

We now define the notion of Well-Quasi Ordering (WQO), which is useful for the proof.

Well-Quasi Ordering and Higman's Lemma. We fix a set X and a relation $\leq$ over X. $\leq$ is said to be a *preorder/quasi order* if it is reflexive ($\forall u \in X : u \leq u$), and transitive ($\forall u, v, w \in X : u \leq v$ and $v \leq w$ implies $u \leq w$). A preorder is a *well-quasi order* (WQO) iff for every infinite sequence $u_1 u_2 \ldots$ over X, there exists a pair $i < j$ such that $u_i \leq u_j$. Let $\preceq$ be an order relation over sequences of X (that is, over X^*) defined as follows. We write $u_1 u_2 \ldots u_m \preceq u'_1 u'_2 \ldots u'_n$ if there exists a strictly increasing function from $\kappa : \{1, \ldots m\} \mapsto \{1, \ldots n\}$ such that for all $i \in \{1, \ldots, m\}$, $u_i \leq u'_{\kappa(i)}$. With this notation, we can phrase Higman's lemma as follows.

▶ **Lemma 5.1** (Higman's Lemma [22]). *If $\leq$ is a WQO over X, then $\preceq$ is a WQO over X^*.*

Run embeddings. The proof of Theorem 1.2 will employ the concept of run embeddings, which was introduced by Jančar [27] and Leroux [30]. Towards its definition, we first define an ordering on steps. Given steps $s = ((p, \boldsymbol{u}), \boldsymbol{a}, (q, \boldsymbol{v}))$ and $s' = ((p', \boldsymbol{u}'), \boldsymbol{a}', (q', \boldsymbol{v}'))$, we write $s \leq s'$ if $p' = p$, $q' = q$, $\boldsymbol{a}' = \boldsymbol{a}$, and $\boldsymbol{u} \leq \boldsymbol{u}'$. Here, $\boldsymbol{u} \leq \boldsymbol{u}'$ means that $\boldsymbol{u}(i) \leq \boldsymbol{u}'(i)$ for every $i \in \{1, \ldots, d\}$. Note that $s \leq s'$ implies $\boldsymbol{v} \leq \boldsymbol{v}'$.

The ordering on steps now induces an embedding ordering on runs. Suppose $\rho = s_1 \cdots s_m$ and $\rho' = s'_1 \cdots s'_n$ are runs. An *embedding of ρ in ρ'* is a strictly monotone map $\sigma \colon \{1, \ldots, m\} \to \{1, \ldots, n\}$ such that (i) $\sigma(1) = 1$ and (ii) $s_i \leq s'_{\sigma(i)}$ for every $i \in \{1, \ldots, m\}$.

Run amalgamation. A key property of the run embedding is the *amalgamation property* as observed by Leroux and Schmitz [34]. Aside from our paper, amalgamation of runs has been used in several other works for constructing runs in VASS and related models [2,10,12,33]. If $\boldsymbol{c} = (p, \boldsymbol{u}) \in Q \times \mathbb{N}^d$ is a configuration of a VASS and $\boldsymbol{w} \in \mathbb{N}^d$ is a vector, then by $\boldsymbol{c} + \boldsymbol{w}$ we denote the configuration $(p, \boldsymbol{u} + \boldsymbol{w})$. If σ is an embedding of ρ in ρ', then we can define a new run as follows. First, we write ρ as $\boldsymbol{c}_0 \xrightarrow{t_1} \boldsymbol{c}_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \boldsymbol{c}_n$ with configurations $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_n$ and transitions $t_1, \ldots, t_n$. Then since σ is an embedding, we can write ρ' as

$$\boldsymbol{c}_0 + \boldsymbol{w}_0 \xrightarrow{\tau_0} \boldsymbol{c}_0 + \boldsymbol{w}_1 \xrightarrow{t_1} \boldsymbol{c}_1 + \boldsymbol{w}_1 \xrightarrow{\tau_1} \boldsymbol{c}_1 + \boldsymbol{w}_2 \xrightarrow{t_2} \boldsymbol{c}_2 + \boldsymbol{w}_2 \xrightarrow{\tau_2} \\ \cdots \xrightarrow{\tau_{n-1}} \boldsymbol{c}_{n-1} + \boldsymbol{w}_k \xrightarrow{t_n} \boldsymbol{c}_n + \boldsymbol{w}_n \xrightarrow{\tau_n} \boldsymbol{c}_n + \boldsymbol{w}_{n+1}$$

where $\tau_0, \ldots, \tau_n$ are transition sequences and $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{n+1}$ are vectors in $\mathbb{N}^d$ with $\boldsymbol{w}_0 = \boldsymbol{0}$. Now we define a new run ρ'^{σ} as

$$\begin{aligned} \boldsymbol{c}_0 + \boldsymbol{w}_0 &\xrightarrow{\tau_0} \boldsymbol{c}_0 + \boldsymbol{w}_1 \xrightarrow{\tau_0} \boldsymbol{c}_0 + \boldsymbol{w}_1 + \boldsymbol{w}_1 \xrightarrow{t_1} \boldsymbol{c}_1 + \boldsymbol{w}_1 + \boldsymbol{w}_1 \\ &\xrightarrow{\tau_1} \boldsymbol{c}_1 + \boldsymbol{w}_1 + \boldsymbol{w}_2 \xrightarrow{\tau_1} \boldsymbol{c}_1 + \boldsymbol{w}_2 + \boldsymbol{w}_2 \xrightarrow{t_2} \boldsymbol{c}_2 + \boldsymbol{w}_2 + \boldsymbol{w}_2 \\ &\vdots \\ &\xrightarrow{\tau_{n-1}} \boldsymbol{c}_{n-1} + \boldsymbol{w}_{n-1} + \boldsymbol{w}_n \xrightarrow{\tau_{n-1}} \boldsymbol{c}_{n-1} + \boldsymbol{w}_n + \boldsymbol{w}_n \xrightarrow{t_n} \boldsymbol{c}_n + \boldsymbol{w}_n + \boldsymbol{w}_n \\ &\xrightarrow{\tau_n} \boldsymbol{c}_n + \boldsymbol{w}_{n+1} + \boldsymbol{w}_n \xrightarrow{\tau_n} \boldsymbol{c}_n + \boldsymbol{w}_{n+1} + \boldsymbol{w}_{n+1}. \end{aligned}$$

Thus, the action sequence of this run is $\tau_0\tau_0 t_1 \tau_1 \tau_1 \cdots t_n \tau_n \tau_n$. Of course, this process can be repeated. If we do this m times, we obtain a run with the action sequence $\tau_0^m t_1 \tau_1^m \cdots t_n \tau_n^m$.

If there exists an embedding of a run ρ into ρ', then we write $\rho \trianglelefteq \rho'$. Since the ordering $\leq$ on steps is a well-quasi ordering and $\trianglelefteq$ is just the embedding relation induced by $\leq$, the following is a direct consequence of Higman's Lemma (Lemma 5.1):

▶ **Lemma 5.2.** *On the set of runs of $\mathcal{V}$, the ordering $\trianglelefteq$ is a well-quasi ordering.*

Reversal increasing embeddings. If ρ and ρ' are as above and σ is an embedding of ρ into ρ', then we say that σ is *reversal increasing* if for some $i \in [0, n]$, the action sequence τ_i contains at least one reversal. We use this notion to prove the first step of Theorem 1.2.

▶ **Lemma 5.3.** *Let $f \colon \mathbb{N} \to \mathbb{N}$ be a monotone function that is not essentially bounded for VASS. Then there is a constant $c \in \mathbb{N}$ such that $f(n) \geq (n - c)/c$ for every $n \in \mathbb{N}$, $n \geq c$.*

Proof. Let $f \colon \mathbb{N} \to \mathbb{N}$ be any monotone function which is not essentially bounded for VASS. Then there exists a VASS $\mathcal{V}$ that is f-reversal bounded, but not reversal bounded.

We first claim that then there exist two runs ρ and ρ' such that ρ embeds into ρ' via a reversal increasing embedding. Towards a contradiction, suppose there is no reversal increasing embedding between runs. Since $\trianglelefteq$ is a well-quasi ordering, the set of runs has a finite set $\{\rho_1, \ldots, \rho_r\}$ of minimal runs. Let $R \in \mathbb{N}$ be the maximal number of reversals within the runs $\rho_1, \ldots, \rho_r$. Since every run of $\mathcal{V}$ embeds at least one of the minimal runs $\rho_1, \ldots, \rho_r$ and every embedding is not reversal increasing, every run of $\mathcal{V}$ has at most r reversals. Thus, f is essentially bounded, against our assumption. This proves our claim. Thus, we have runs ρ and ρ' such that ρ embeds into ρ' via some reversal increasing embedding σ.

Let us now show that f belongs to $\Omega(n)$. Suppose ρ and ρ' are our constructed runs such that ρ has action sequence $t_1 \cdots t_N$. Moreover, for each $m \in \mathbb{N}$, let $\tau_0^m t_1 \tau_1^m \cdots t_N \tau_N^m$ be the action sequences resulting from amalgamating ρ and ρ' exactly m times. Since σ is reversal increasing, observe that $\tau_0^m t_1 \tau_1^m \cdots t_N \tau_N^m$ has at least m reversals. Moreover, if we set $e := |\tau_0| + \cdots + |\tau_N|$, then the length of $\tau_0^m t_1 \tau_1^m \cdots t_N \tau_N^m$ is $N + m \cdot e$.

We have thus constructed, for every $m \in \mathbb{N}$, a run of length $N + m \cdot e$ with at least m reversals. We now claim that with $c := e + N$, we have indeed $f(n) \geq (n - c)/c$ for every $n \in \mathbb{N}$, $n \geq c$. Our constructed runs yield $f(N + me) \geq m$ for every $m \geq 1$. Therefore, if $n > N$ and $n \equiv N \pmod{e}$, then $f(n) \geq (n - N)/e$. Now for every $n \in \mathbb{N}$, $n \geq c$, we can pick $i \in [1, e]$ such that $n - i \equiv N \pmod{e}$ and thus $f(n - i) \geq (n - i - N)/e$. Since f is monotone, this implies $f(n) \geq f(n - i) \geq (n - i - N)/e \geq (n - c)/c$, as claimed. ◀

Proof of Theorem 1.2. Suppose $f \colon \mathbb{N} \to \mathbb{N}$ is a monotone function that is not essentially bounded for VASS. According to Lemma 5.3, there is a constant $c \in \mathbb{N}$ such that $f(n) \geq (n - c)/c$ for every $n \in \mathbb{N}$, $n \geq c$. This already implies that f belongs to $\Omega(n)$. It remains to argue that $\mathsf{Reach}_{\mathsf{VASS}}(f)$ is Ackermann-hard. This is simple: Given a VASS $\mathcal{V}_0$, we turn it into a VASS $\mathcal{V}_1$ that begins by taking c empty steps, and afterwards, it simulates each step of $\mathcal{V}_0$ by first taking c empty steps (i.e. steps that do not change any counters). Then $\mathcal{V}_1$ is clearly f-reversal-bounded, because in any run of length n, there are at most $(n - c)/c$ steps that add a non-zero vector. Since reachability in VASS is Ackermann-complete [11, 32, 35], this shows that reachability in f-reversal-bounded VASS is also Ackermann-complete. ◀

References

1. Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Philippe Schnoebelen. Flat acceleration in symbolic model checking. In Doron A. Peled and Yih-Kuen Tsay, editors, *Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005, Proceedings*, volume 3707 of *Lecture Notes in Computer Science*, pages 474–488. Springer, 2005. doi:10.1007/11562948_35.
2. Pascal Baumann, Flavio D'Alessandro, Moses Ganardi, Oscar H. Ibarra, Ian McQuillan, Lia Schütze, and Georg Zetzsche. Unboundedness problems for machines with reversal-bounded counters. In Orna Kupferman and Pawel Sobocinski, editors, *Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings*, volume 13992 of *Lecture Notes in Computer Science*, pages 240–264. Springer, 2023. doi:10.1007/978-3-031-30829-1_12.
3. Marcello M. Bersani and Stéphane Demri. The complexity of reversal-bounded model-checking. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Frontiers of Combining Systems, 8th International Symposium, FroCoS 2011, Saarbrücken, Germany, October 5-7, 2011. Proceedings*, volume 6989 of *Lecture Notes in Computer Science*, pages 71–86. Springer, 2011. doi:10.1007/978-3-642-24364-6_6.
4. Alin Bostan, Arnaud Carayol, Florent Koechlin, and Cyril Nicaud. Weakly-unambiguous Parikh automata and their link to holonomic series. 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 114:1–114:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.114.
5. Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Affine Parikh automata. *RAIRO Theor. Informatics Appl.*, 46(4):511–545, 2012. doi:10.1051/ita/2012013.
6. Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Bounded Parikh automata. *Int. J. Found. Comput. Sci.*, 23(8):1691–1710, 2012. doi:10.1142/S0129054112400709.
7. Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Unambiguous constrained automata. *Int. J. Found. Comput. Sci.*, 24(7):1099–1116, 2013. doi:10.1142/S0129054113400339.
8. Michaël Cadilhac, Andreas Krebs, and Pierre McKenzie. The algebraic theory of Parikh automata. *Theory Comput. Syst.*, 62(5):1241–1268, 2018. doi:10.1007/s00224-017-9817-2.

9 Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Regular separability of Parikh automata. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 117:1–117:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.117.

10 Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Separability of reachability sets of vector addition systems. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 24:1–24:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.24.

11 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.

12 Wojciech Czerwinski and Georg Zetzsche. An approach to regular separability in vector addition systems. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 341–354. ACM, 2020. doi:10.1145/3373718.3394776.

13 Zhe Dang, Oscar H. Ibarra, and Pierluigi San Pietro. Liveness verification of reversal-bounded multicounter machines with a free counter. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, 21st Conference, Bangalore, India, December 13-15, 2001, Proceedings*, volume 2245 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 2001. doi:10.1007/3-540-45294-X_12.

14 Stéphane Demri and Arnaud Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In C.-H. Luke Ong, editor, *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6014 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2010. doi:10.1007/978-3-642-12032-9_13.

15 Alain Finkel and Arnaud Sangnier. Reversal-bounded counter machines revisited. In Edward Ochmanski and Jerzy Tyszkiewicz, editors, *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, volume 5162 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2008. doi:10.1007/978-3-540-85238-4_26.

16 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

17 Mario Grobler, Leif Sabellek, and Sebastian Siebertz. Remarks on Parikh-recognizable omega-languages, 2023. arXiv:2307.07238.

18 Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. Parikh automata over infinite words. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 40:1–40:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.40.

19 Eitan M. Gurari and Oscar H. Ibarra. The complexity of decision problems for finite-turn multicounter machines. In Shimon Even and Oded Kariv, editors, *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*, volume 115 of *Lecture Notes in Computer Science*, pages 495–505. Springer, 1981. doi:10.1007/3-540-10843-2_39.

20 Christoph Haase. Subclasses of Presburger arithmetic and the weak EXP hierarchy. 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 47:1–47:10. ACM, 2014. doi:10.1145/2603088.2603092.

21 Matthew Hague and Anthony Widjaja Lin. Model checking recursive programs with numeric data types. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification – 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 743–759. Springer, 2011. doi:10.1007/978-3-642-22110-1_60.

22 Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(1):326–336, 1952.

23 John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159, 1979. doi:10.1016/0304-3975(79)90041-0.

24 Oscar H Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM (JACM)*, 25(1):116–133, 1978.

25 Oscar H. Ibarra, Jianwen Su, Zhe Dang, Tevfik Bultan, and Richard A. Kemmerer. Conter machines: Decidable properties and applications to verification problems. In Mogens Nielsen and Branislav Rovan, editors, *Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Bratislava, Slovakia, August 28 – September 1, 2000, Proceedings*, volume 1893 of *Lecture Notes in Computer Science*, pages 426–435. Springer, 2000. doi:10.1007/3-540-44612-5_38.

26 Matthias Jantzen and Alexy Kurganskyy. Refining the hierarchy of blind multicounter languages and twist-closed trios. *Inf. Comput.*, 185(2):159–181, 2003. doi:10.1016/S0890-5401(03)00087-7.

27 Petr Jančar. Decidability of a temporal logic problem for Petri nets. *Theor. Comput. Sci.*, 74(1):71–93, 1990. doi:10.1016/0304-3975(90)90006-4.

28 Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 – July 4, 2003. Proceedings*, volume 2719 of *Lecture Notes in Computer Science*, pages 681–696. Springer, 2003. doi:10.1007/3-540-45061-0_54.

29 Michel Latteux. Cônes rationnels commutatifs. *J. Comput. Syst. Sci.*, 18(3):307–333, 1979. doi:10.1016/0022-0000(79)90039-4.

30 Jérôme Leroux. Vector addition system reachability problem: a short self-contained proof. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 307–316. ACM, 2011. doi:10.1145/1926385.1926421.

31 Jérôme Leroux. Presburger vector addition systems. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 23–32. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.7.

32 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

33 Jérôme Leroux, M. Praveen, Philippe Schnoebelen, and Grégoire Sutre. On functions weakly computable by pushdown Petri nets and related systems. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:15)2019.

34 Jérôme Leroux and Sylvain Schmitz. Demystifying reachability in vector addition systems. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 56–67. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.16.

35 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785796.

36 Jérôme Leroux and Grégoire Sutre. Flat counter automata almost everywhere! In Doron A. Peled and Yih-Kuen Tsay, editors, *Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005, Proceedings*, volume 3707 of *Lecture Notes in Computer Science*, pages 489–503. Springer, 2005. doi:10.1007/11562948_36.

37 Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961. URL: http://www.jstor.org/stable/1970290.

38 Loic Pottier. Minimal solutions of linear Diophantine systems: Bounds and algorithms. In Ronald V. Book, editor, *Rewriting Techniques and Applications, 4th International Conference, RTA-91, Como, Italy, April 10-12, 1991, Proceedings*, volume 488 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 1991. doi:10.1007/3-540-53904-2_94.

39 Georg Zetzsche. The complexity of downward closure comparisons. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 123:1–123:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.123.

Perfect Matchings and Popularity in the Many-To-Many Setting

Telikepalli Kavitha
Tata Institute of Fundamental Research, Mumbai, India

Kazuhisa Makino
Research Institute for Mathematical Sciences, Kyoto University, Japan

Abstract

We consider a matching problem in a bipartite graph G where every vertex has a capacity and a strict preference list ranking its neighbors. We assume that G admits a perfect matching, i.e., one that fully matches all vertices. It is only perfect matchings that are feasible here and we seek one that is popular within the set of perfect matchings – it is known that such a matching exists in G and can be efficiently computed. Now we are in the weighted setting, i.e., there is a cost function on the edge set, and we seek a min-cost popular perfect matching in G. We show that such a matching can be computed in polynomial time.

Our main technical result shows that every popular perfect matching in a hospitals/residents instance G can be realized as a popular perfect matching in the marriage instance obtained by cloning vertices. Interestingly, it is known that such a mapping does not hold for popular matchings in a hospitals/residents instance.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Bipartite graphs, Matchings under preferences, Capacities, Dual certificates

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.43

Funding *Telikepalli Kavitha*: Supported by the Department of Atomic Energy, Government of India, under project no. RTI4001. This work was done while visiting RIMS, Kyoto.
Kazuhisa Makino: Partially supported by JST ERATO Grant Number JPMJER2301, JSPS KAKENHI Grant Numbers JP20H05967, JP19K22841, and JP20H00609, and the joint project of Kyoto University and Toyota Motor Corporation, titled "Advanced Mathematical Science for Mobility Society".

Acknowledgements Thanks to Naoyuki Kamiyama for asking us about the computational complexity of the min-cost popular maximum matching problem when vertices have capacities.

1 Introduction

We consider a matching problem in a bipartite graph $G = (R \cup H, E)$ where R is a set of residents and H is a set of hospitals. Every resident r seeks to be matched to a hospital while every hospital h has an integral capacity $\mathsf{cap}(h) \geq 1$ and seeks to be matched to $\mathsf{cap}(h)$ many residents in any matching.

▶ **Definition 1.** *A matching M in $G = (R \cup H, E)$ is a subset of E such that $|M(r)| \leq 1$ for each $r \in R$ and $|M(h)| \leq \mathsf{cap}(h)$ for each $h \in H$, where $M(v) = \{u : (u, v) \in M\}$ for any $v \in R \cup H$.*

Every vertex $v \in R \cup H$ has a strict ranking of its neighbors. Such a graph G is called a hospitals/residents instance and this is a well-studied model in two-sided matching markets. This model includes several real-world applications such as matching doctors to residencies in hospitals [9, 29] and students to schools and colleges [1, 4]. It is crucial to match as many doctors as possible to residencies and all students to schools and colleges.

© Telikepalli Kavitha and Kazuhisa Makino;
licensed under Creative Commons License CC-BY 4.0
43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).
Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 43; pp. 43:1–43:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Let us assume that the input instance admits a perfect matching, i.e., one that fully matches all vertices in $R \cup H$. Among other scenarios, such a model occurs in the "lab rotation" problem where each student gets assigned to a lab during training. At certain intervals, based on the preferences of students and those in charge of labs, there may be a cyclic shift of students among labs. The set of feasible solutions is the set of perfect matchings. Note that the instance admits a perfect matching which is the current assignment of students to labs. We seek a "best perfect matching" as per vertex preferences – in the domain of matchings under preferences, *stable* matchings are usually regarded as the best matchings.

Stable matchings. A matching M is stable if there is no edge that *blocks* M, where an edge (r, h) blocks M if (i) either r is unmatched in M or r prefers h to its partner in M and (ii) either h has less than $\mathsf{cap}(h)$ partners in M or h prefers r to its worst partner in M. Stable matchings always exist in G and the classical Gale-Shapley algorithm [14] finds one. Stability is a strong and rather restrictive notion – it is known that every stable matching matches the same subset of residents and every hospital gets matched to the same capacity; this is known as the "Rural Hospitals Theorem" [15, 31].

Stable matchings need not have maximum cardinality. Consider the following instance where $R = \{r, r'\}, H = \{h, h'\}$, and all vertex capacities are 1. The preferences of residents are as follows: $r : h \succ h'$ and $r' : h$, i.e., r's top choice is h and second choice is h' while r' has only one neighbor h. Similarly, the preferences of hospitals are as follows: $h : r \succ r'$ and $h' : r$. This instance has only one stable matching $S = \{(r, h)\}$ that leaves r' and h' unmatched. Observe that this instance has a perfect matching $M = \{(r, h'), (r', h)\}$ that matches all the residents and hospitals.

A perfect matching with the least number of blocking edges would be a natural relaxation of stability; unfortunately, finding such a matching is NP-hard [6]. A well-studied relaxation of stability that offers a meaningful and tractable solution to the problem of finding a "best perfect matching" is the notion of *popularity*.

Popularity. Popularity is based on voting by vertices on feasible matchings. In the one-to-one or marriage setting, the preferences of a vertex over its neighbors extend naturally to preferences over matchings – so every vertex orders feasible matchings in the order of its partners in these matchings and being unmatched is the worst option for any vertex. Popular matchings are (weak) *Condorcet winners* [12, 24] in the above voting instance where all matchings are feasible, i.e., a popular matching M does not lose a head-to-head election against any matching N where each vertex either casts a vote for the matching in $\{M, N\}$ that it prefers or it abstains from voting if its assignment is the same in M and N. There need not be a weak Condorcet winner in a typical voting instance; however, in the context of matchings in the one-to-one or marriage setting, every stable matching is a weak Condorcet winner [16] – so popular matchings always exist in this setting.

Recall that we are in the many-to-one or hospitals/residents setting, i.e., hospitals have capacities. So we need to specify how a hospital votes over different subsets of its neighbors. Thus for a hospital h, we need to compare two subsets $M(h)$ and $N(h)$ of $\mathsf{Nbr}(h)$, where $\mathsf{Nbr}(h)$ is the set of neighbors of h. We will follow the method from [8] for this comparison. The definition in [8] of voting by a hospital h between two subsets of $\mathsf{Nbr}(h)$ is the following:

- First, all residents that are contained in both sets are removed and then a bijection from the first set to the second set is determined.[1] Every vertex is compared with its image under this bijection, thus the vote depends on the bijection that is chosen.

[1] If the sets are not of equal size, then dummy vertices that are less preferred to all non-dummy vertices are added to the smaller set.

Voting by vertices. Let us first formally define how a resident r casts its vote between two neighbors h and h'. The function $\mathsf{vote}_r(h, h')$ is 1 if r prefers h to h', it is -1 if r prefers h' to h, and it is 0 otherwise, i.e., $h = h'$. The function $\mathsf{vote}_r(\cdot, \cdot)$ that compares two neighbors of r extends naturally to any pair of matchings M and N as $\mathsf{vote}_r(M, N) = \mathsf{vote}_r(M(r), N(r))$. Note that $M(r)$ (resp., $N(r)$) is null if r is unmatched in M (resp., N) and this is the worst state for any vertex. Every resident casts a vote in $\{0, \pm 1\}$ in the M-vs-N election.

A hospital h with capacity $\mathsf{cap}(h)$ is allowed to cast up to $\mathsf{cap}(h)$ many votes. Let S and T be two subsets of $\mathsf{Nbr}(h)$; the vertex h compares these two subsets as follows:

- a bijection ψ is chosen from $S' = S \setminus T$ to $T' = T \setminus S$;
- every resident $r \in S'$ is compared with $\psi(r) \in T'$;
- the number of wins minus the number of losses is h's vote for S versus T.

The bijection ψ that is chosen will be the one that *minimizes* h's vote for S' versus T'. More formally, the vote of h for S versus T, denoted by $\mathsf{vote}_h(S, T)$, is defined as follows where $|S'| = |T'| = k$ and $\Pi[k]$ is the set of permutations on $\{1, \ldots, k\}$:

$$\mathsf{vote}_h(S, T) = \min_{\sigma \in \Pi[k]} \sum_{i=1}^{k} \mathsf{vote}_h(S'[i], T'[\sigma(i)]), \tag{1}$$

where $S'[i]$ is the i-th ranked resident in S' and $T'[\sigma(i)]$ is the $\sigma(i)$-th ranked resident in T'.

For any pair of matchings M and N in G, let $\mathsf{vote}_h(M, N) = \mathsf{vote}_h(S, T)$, where $S = M(h)$ and $T = N(h)$. So $\mathsf{vote}_h(M, N)$ counts the number of votes by h for $M(h)$ versus $N(h)$ when the two sets $M(h) \setminus N(h)$ and $N(h) \setminus M(h)$ are compared in the order that is most adversarial or *negative* for M. The two matchings M and N are compared using $\Delta(M, N) = \sum_{v \in R \cup H} \mathsf{vote}_v(M, N)$. A matching M is popular if $\Delta(M, N) \geq 0$ for all matchings N in G.

Recall our assumption that the set of feasible solutions for us is the set of perfect matchings. Hence the matchings of interest to us are *popular perfect matchings*, defined below.

▶ **Definition 2.** *A perfect matching M is a popular perfect matching if $\Delta(M, N) \geq 0$ for all perfect matchings N in G.*

It was shown in [22] that there always exists a popular maximum matching (i.e., a maximum matching that is popular within the set of maximum matchings) in the one-to-one setting; moreover, such a matching can be computed in polynomial time. Thus whenever the input instance admits a perfect matching, there always exists a popular perfect matching. This result from the one-to-one setting extends easily to the many-to-one or hospitals/residents setting.

Let G' be the marriage instance obtained from G by making $\mathsf{cap}(h)$ many *clones* $h_1, \ldots, h_{\mathsf{cap}(h)}$ of each hospital h where $h_1 \succ_r \cdots \succ_r h_{\mathsf{cap}(h)}$ for each resident r adjacent to h in G (Section 1.2 has more details). There is a natural map f from the set of popular perfect matchings in G' to the set of popular perfect matchings in G, where for any popular perfect matching M' in the marriage instance G', the many-to-one matching $f(M') = M$ is obtained by identifying all the clones of the same hospital. It is not difficult to show that M is a popular perfect matching in the hospitals/residents instance G (see Section 3).

Min-cost popular perfect matchings. There might be exponentially many popular perfect matchings in G, hence we would like to find an *optimal* one. Let us assume there is a function $\mathsf{cost} : E \to \mathbb{R}$, so the cost of a matching M is $\sum_{e \in M} \mathsf{cost}(e)$. Thus what we seek is a min-cost popular perfect matching in G. This is a natural problem in discrete optimization.

Solving the min-cost popular perfect matching problem efficiently implies efficient algorithms for several desirable popular perfect matching problems such as finding one with the highest utility when every edge (r, h) has an associated utility or one with forced/forbidden edges or one which optimizes the worst rank of a hospital that any resident gets matched to. Observe that via appropriate cost functions, we can "access" the entire set of popular perfect matchings.

A polynomial time algorithm for computing a min-cost popular maximum matching in the one-to-one setting is known [23]. However no polynomial time algorithm is currently known for this problem in the many-to-one setting. Recall the natural map f described earlier. Is $f : \{\text{popular perfect matchings in } G'\} \to \{\text{popular perfect matchings in } G\}$ surjective? If so, then we can obtain a min-cost popular perfect matching in G by computing a min-cost popular perfect matching in G'.

However there is no a priori reason to believe this map to be surjective. In fact, it is known that the natural map from {popular matchings in G'} to {popular matchings in G} is *not* surjective. The following instance $G = (R \cup H, E)$ was given in [8]: $R = \{p, q, r, s\}$ and $H = \{h, h', h''\}$ where $\mathsf{cap}(h) = 2$ and $\mathsf{cap}(h') = \mathsf{cap}(h'') = 1$. The preference orders of vertices are as follows:

$p\colon h \succ h''$	$r\colon h$	$h\colon p \succ q \succ r \succ s$	$h''\colon p$
$q\colon h \succ h'$	$s\colon h$	$h'\colon q$	

The marriage instance G' has vertex set $R = \{p, q, r, s\}$ and $H' = \{h_1, h_2, h', h''\}$ where h_1 and h_2 are the two clones of h. The preference order of p in G' is $h_1 \succ h_2 \succ h''$ and so on.

The matching $N = \{(p, h), (q, h'), (r, h)\}$ is popular in G. The two possible realizations of N in G' are $N_1 = \{(p, h_1), (q, h'), (r, h_2)\}$ and $N_2 = \{(p, h_2), (q, h'), (r, h_1)\}$. It was shown in [8] that neither N_1 nor N_2 is popular in G'.

1. The matching $M_1 = \{(p, h''), (q, h_2), (r, h_1)\}$ is more popular than N_1 since the four vertices q, r, h_2, h'' prefer M_1 to N_1 while the three vertices p, h_1, h' prefer N_1 to M_1 and s is indifferent between M_1 and N_1. Thus N_1 is not a popular matching in G'.
2. Similarly, the matching $M_2 = \{(p, h_1), (q, h'), (s, h_2)\}$ is more popular than N_2 since the three vertices p, h_1, s prefer M_2 to N_2 while the two vertices r and h_2 prefer N_2 to M_2 and the vertices q, h', h'' are indifferent between M_2 and N_2. Thus N_2 is not a popular matching in G'. Hence neither N_1 nor N_2 is popular in G'.

- Observe that N is not a perfect matching in G. Do popular *perfect matchings* exhibit the same behavior? That is, is there a hospitals/residents instance G where the map $f : \{\text{popular perfect matchings in } G'\} \to \{\text{popular perfect matchings in } G\}$ is not surjective?

We show the following result. In contrast to the behavior of popular matchings, every popular perfect matching in G has a preimage (under f) as a popular perfect matching in G'. Hence solving the min-cost popular perfect matching in the marriage instance G' solves the min-cost popular perfect matching problem in the hospitals/residents instance G. Thus, by using the polynomial time algorithm to find a min-cost popular maximum matching in G' [23], a min-cost popular perfect matching in G can be determined in polynomial time.

▶ **Theorem 3.** *Let $G = (R \cup H, E)$ be a hospitals/residents instance with $\mathsf{cost} : E \to \mathbb{R}$, where hospitals have capacities and every vertex has a strict ranking of its neighbors. If G admits a perfect matching then a min-cost popular perfect matching in G can be computed in polynomial time.*

Many-to-many matchings. We study the min-cost popular perfect matching problem in the many-to-many setting as well. As before, every hospital h has a capacity $\mathsf{cap}(h) \geq 1$ and now every resident (or doctor) has a capacity $\mathsf{cap}(r) \geq 1$, i.e., a doctor can be associated with more than one hospital.

▶ **Definition 4.** *A matching M in $G = (R \cup H, E)$ is a subset of E such that $|M(v)| \leq \mathsf{cap}(v)$ for each $v \in R \cup H$, where $M(v) = \{u : (u, v) \in M\}$ for any $v \in R \cup H$.*

Though vertices in both R and H have capacities, since M is a set (and not a multiset), M can contain at most one copy of any edge (r, h). In order to compare any pair of matchings M and N, every vertex $v \in R \cup H$ casts up to $\mathsf{cap}(v)$ many votes as given by $\mathsf{vote}_v(M, N)$ (see Eq. (1) in Section 1).

As before, we are only interested in perfect matchings, in particular, in *popular perfect matchings* (see Definition 2). Whenever the input instance G admits a perfect matching, popular perfect matchings exist in the many-to-many setting – any popular perfect matching in the corresponding marriage instance $G' = (R' \cup H', E')$ maps to a popular perfect matching in the original many-to-many matching instance $G = (R \cup H, E)$.[2]

Now the input also consists of a function $\mathsf{cost} : E \to \mathbb{R}$ and what we seek is a min-cost popular perfect matching in G. The question again is whether the natural map f from the set of popular perfect matchings in G' to the set of popular perfect matchings in G is surjective or not. As in the many-to-one setting, we show the function f is surjective and this leads to the following result.

▶ **Theorem 5.** *Let $G = (R \cup H, E)$ be a many-to-many matching instance with $\mathsf{cost} : E \to \mathbb{R}$, where vertices have capacities and every vertex has a strict ranking of its neighbors. If G admits a perfect matching then a min-cost popular perfect matching in G can be computed in polynomial time.*

1.1 Background

Algorithmic questions in popular matchings were first studied in the domain of one-sided preference lists in bipartite graphs where it is only vertices on the left or *agents* that have preferences [2]; vertices on the right are *objects* with no preferences. Popular matchings need not always exist here and a polynomial time algorithm was given in [2] to determine if a given instance admits a popular matching and find one, if so.

The notion of popularity was proposed by Gärdenfors [16] in 1975 in the domain of two-sided strict preferences or the stable marriage problem (i.e., in the one-to-one setting) where he observed that stable matchings are popular. So popular matchings always exist in this setting. It was shown in [5, 11] that when preferences include ties (even one-sided ties), it is NP-hard to decide if a popular matching exists or not. It was shown in [19] that every stable matching in a marriage instance is a min-size popular matching. Polynomial time algorithms to find a max-size popular matching were shown in [19, 22].

As mentioned earlier, it was shown in [22] that popular maximum matchings always exist in a marriage instance. It was shown there that any stable matching in an auxiliary instance is a popular maximum matching in the given marriage instance. More recently, the converse was shown in [23], i.e., a matching M in a marriage instance is a popular maximum matching if and only if M can be realized as a stable matching in this auxiliary instance. This yielded

[2] Each vertex v in G is replaced by $\mathsf{cap}(v)$ many clones $v_1, \ldots, v_{\mathsf{cap}(v)}$ in the marriage instance G' and $v_1 \succ_{u_i} \cdots \succ_{u_i} v_{\mathsf{cap}(v)}$ for each neighbor u_i in G'.

a polynomial time algorithm for the min-cost popular maximum matching problem in a marriage instance. Though there is a polynomial time algorithm for computing a min-cost popular maximum matching in a marriage instance, finding a min-cost popular matching is NP-hard [13]. We refer to [10] for a survey on results in popular matchings in the marriage setting.

The stable matching problem has been extensively studied in the hospitals/residents setting and also in the many-to-many setting [3, 7, 18, 17, 20, 21, 30, 33] and a min-cost stable matching in a hospitals/residents instance can be computed in polynomial time [32, 34]. The notion of popularity was extended from the marriage setting to the many-to-many setting in [8] and [28], independently. A polynomial time algorithm to compute a max-size popular matching in the many-to-many setting was given in [8]. It was also shown in [8] that every stable matching in the many-to-many setting is popular; so though a rather strong definition of popularity was adopted here, popular matchings always exist. The definition of popularity considered in [28] is weaker than the one in [8]; in order to compare a pair of matchings M_0 and M_1, every hospital h uses the bijection that compares the top neighbor in $M_0(h) \setminus M_1(h)$ with the top neighbor in $M_1(h) \setminus M_0(h)$, and so on, i.e., the permutation σ in Eq. (1) is the identity permutation.

Popular matchings where vertices have capacity lower bounds have been studied in the hospitals/residents setting and in the many-to-many setting [25, 26, 27], where it is only matchings that satisfy these lower bounds that are feasible. It was shown in these works that popular feasible matchings always exist and can be computed in polynomial time.

1.2 Our Techniques

For any hospitals/residents instance $G = (R \cup H, E)$ with strict preferences, there is a corresponding marriage instance G', in other words, each vertex in G' has capacity 1. The vertex set of G' is $R \cup H'$ where $H' = \cup_{h \in H}\{h_1, h_2, \ldots, h_{\mathsf{cap}(h)}\}$.

The set of neighbors of each h_i in G' is exactly the same as the set of h's neighbors in G. So $G' = (R \cup H', E')$ is a one-to-one or marriage instance where $E' = \{(r, h_i) : (r, h) \in E$ and $1 \leq i \leq \mathsf{cap}(h)\}$. Every vertex in G' has a strict preference order over its neighbors.

- For $h \in H$ and $1 \leq i \leq \mathsf{cap}(h)$: the preference order of h_i in G' is exactly the same as h's preference order in G.
- For $r \in R$: the preference order of r in G' is the same as r's preference order in G where every neighbor h in G gets replaced by all its clones in the order $h_1 \succ h_2 \succ \cdots \succ h_{\mathsf{cap}(h)}$. So if r's preference order in G is $h \succ h'$ then r's preference order in G' is:

$$h_1 \succ h_2 \succ \cdots \succ h_{\mathsf{cap}(h)} \succ h'_1 \succ h'_2 \succ \cdots \succ h'_{\mathsf{cap}(h')}.$$

Canonical realization. Let M be a perfect (many-to-one) matching in G. For any $h \in H$, recall that $M(h)$ is the set of h's partners in M. The matching M' defined below will be called the *canonical* realization of M in the marriage instance G'.

$$M' = \cup_{h \in H}\{(r_i, h_i) : 1 \leq i \leq \mathsf{cap}(h) \text{ and } r_i \text{ is the } i\text{-th most preferred partner of } h \text{ in } M\}.$$

The canonical realization M' of a popular perfect matching M in the hospitals/residents instance G need *not* be a popular perfect matching in the marriage instance G'. Consider the following example where $R = \{p, q, r\}, H = \{h, h'\}, \mathsf{cap}(h) = 2$, and $\mathsf{cap}(h') = 1$.

$p\colon h \succ h'$ $\qquad$ $h\colon p \succ q \succ r$

$q\colon h \succ h'$ $\qquad$ $h'\colon p \succ q$

$r\colon h$

It is easy to check that $M = \{(p,h),(q,h'),(r,h)\}$ is a popular perfect matching in G. But its canonical realization $M' = \{(p,h_1),(q,h'),(r,h_2)\}$ is not a popular perfect matching in G' since the perfect matching $N = \{(p,h'),(q,h_2),(r,h_1)\}$ in G' is more popular than M'; the four vertices q, r, h_2, h' prefer N to M' while the two vertices p and h_1 prefer M' to N.

Stable matching problems in a hospitals/residents instance G can be easily translated to stable matching problems in the corresponding marriage instance G' since a matching M is stable in G if and only if its canonical realization M' is stable in G'. However popular matchings are more complex than stable matchings as seen in the earlier example (from [8]) where an instance G and a popular matching N were shown such that N has no preimage as a popular matching in G'.

Moreover, as seen in the example above, though M is a popular perfect matching in G, its canonical realization M' need not be a popular perfect matching in G'. Though M' is not a popular perfect matching in G', there is another realization $M'' = \{(p,h_2),(q,h'),(r,h_1)\}$ of M that is a popular perfect matching in G'. So the question that we consider here is the following:

- For any popular perfect matching M in a hospitals/residents instance G, is there always *some* realization of M that is a popular perfect matching in the marriage instance G'?

We show the answer to the above question is "yes". We use LP duality to show this. It is known that every popular perfect matching in a marriage instance G' has a *dual certificate* $\vec{\alpha}$ that certifies its optimality [23] (see Lemma 6 in Section 2). Every popular perfect matching M in a hospitals/residents instance G has a weaker dual certificate $\vec{\gamma}$ that certifies its optimality. We show how to transform the weaker certificate $\vec{\gamma}$ into a stronger certificate $\vec{\alpha}$. This involves obtaining an appropriate realization M'' of M such that M'' is a popular perfect matching in the marriage instance G' with $\vec{\alpha}$ as its dual certificate.

Popular maximum matchings. The popular maximum matching problem generalizes the popular perfect matching problem. In contrast to the above result for popular perfect matchings, it is not the case that for any popular maximum matching M in a hospitals/residents instance G, there always exists some realization of M that is a popular maximum matching in the marriage instance G'. Consider the following example where $R = \{r,s\}, H = \{h,h'\}, \mathsf{cap}(h) = 2$, and $\mathsf{cap}(h') = 1$.

$r\colon h' \succ h$ $\qquad$ $h\colon r \succ s$

$s\colon h' \succ h$ $\qquad$ $h'\colon r \succ s$

It is easy to check that $M = \{(r,h),(s,h')\}$ is a popular maximum matching in G. Note that M is not a perfect matching since $\mathsf{cap}(h) = 2$. We show below that neither the matching $M' = \{(r,h_1),(s,h')\}$ nor the matching $M'' = \{(r,h_2),(s,h')\}$ is a popular maximum matching in G'. So M has no realization as a popular maximum matching in G'.

- The matching M' is more popular than M'' since r and h_1 prefer M' to M'' while h_2 prefers M'' to M' and the vertices s and h' are indifferent between M' and M''.
- The matching $N = \{(r,h'),(s,h_2)\}$ is more popular than M' since the vertices r, h', and h_2 prefer N to M' while the vertices s and h_1 prefer M' to N.

2 Preliminaries

In this section we describe dual certificates for popular perfect matchings in a marriage instance $G_0 = (A \cup B, E_0)$ where vertices have strict preferences. For any matching M in G_0, the following edge weight function wt_M can be defined. For any $e \in E_0$:

$$\text{let } \mathsf{wt}_M(e) = \begin{cases} 2 & \text{if } e \text{ blocks } M; \\ -2 & \text{if the endpoints of } e \text{ prefer their partners in } M \text{ to each other;} \\ 0 & \text{otherwise.} \end{cases}$$

So for any edge $e = (a, b)$, $\mathsf{wt}_M(a, b)$ is the sum of the votes of a and b for each other versus their respective partners in M. Recall that $\mathsf{vote}_a(b, M(a)) \in \{0, \pm 1\}$ and similarly, $\mathsf{vote}_b(a, M(b)) \in \{0, \pm 1\}$. This makes $\mathsf{wt}_M(a, b) \in \{0, \pm 2\}$.

Every popular perfect matching M in a marriage instance $G_0 = (A \cup B, E_0)$ is a max-weight perfect matching in G_0 under the edge weight function wt_M. The linear program LP1 is the max-weight perfect matching LP in G_0 and LP2 is the dual LP.

$$\begin{aligned} \max \sum_{e \in E_0} \mathsf{wt}_M(e) \cdot x_e \qquad & \text{(LP1)} \\ \text{s.t.} \quad \sum_{e \in \delta(v)} x_e = 1 \ \forall\, v \in A \cup B & \\ x_e \geq 0 \ \forall\, e \in E_0. & \end{aligned} \qquad \begin{aligned} \min \sum_{v \in A \cup B} y_v \qquad & \text{(LP2)} \\ \text{s.t.} \quad y_a + y_b \geq \mathsf{wt}_M(a, b) \ \forall\, (a, b) \in E_0. & \end{aligned}$$

For any $v \in A \cup B$, note that $\delta(v)$ is the set of edges incident to v in the graph G_0. The following result (from [23]) shows that LP2 admits a special optimal solution. Here $|A \cup B| = n$ and $|A| = n_0$.

▶ **Lemma 6** ([23]). *Let M be a perfect matching in $G_0 = (A \cup B, E_0)$. Then M is a popular perfect matching in G_0 if and only if there exists $\vec{\alpha} \in \mathbb{R}^n$ that satisfies the following properties:*

1. $\alpha_a \in \{0, -2, -4, \ldots, -2(n_0 - 1)\}$ *for all* $a \in A$.
2. $\alpha_b \in \{0, 2, 4, \ldots, 2(n_0 - 1)\}$ *for all* $b \in B$.
3. $\alpha_a + \alpha_b \geq \mathsf{wt}_M(a, b)$ *for all* $(a, b) \in E_0$.
4. $\sum_{v \in A \cup B} \alpha_v = 0$.

▶ **Definition 7.** *For any popular perfect matching M in a marriage instance G_0, a vector $\vec{\alpha} \in \mathbb{R}^n$ that satisfies properties 1-4 in Lemma 6 will be called a dual certificate for M.*

3 Popular perfect matchings in a hospitals/residents instance

Let $G = (R \cup H, E)$ be a hospitals/residents instance where vertices have strict preferences. We present a characterization of popular perfect matchings in G in Section 3.1 and use this characterization to show helpful dual certificates for popular perfect matchings in Section 3.2.

3.1 A characterization of popular perfect matchings in G

Let M be a perfect matching in G. Recall the *canonical* realization M' of M (from Section 1.2) in the marriage instance $G' = (R \cup H', E')$ corresponding to G. So $M' = \cup_{h \in H} \{(r_i, h_i) : 1 \leq i \leq \mathsf{cap}(h)$, where r_i is the i-th most preferred partner of h in $M\}$.

We know that M' need not be a popular perfect matching in G'. Let $G'_M = (R \cup H', E'_M)$ be a subgraph of G' whose edge set is defined as follows:

$$E'_M = M' \cup \{(r, h_i) : (r, h) \in E \setminus M \text{ and } 1 \le i \le \mathsf{cap}(h)\}.$$

So for any edge $(r, h) \notin M$, all the $\mathsf{cap}(h)$ many copies $(r, h_1), \ldots, (r, h_{\mathsf{cap}(h)})$ of (r, h) are in E'_M while for each edge $(s, h) \in M$, exactly *one* edge, which is the edge in M' (say, (s, h_i)) is in E'_M. Thus $E'_M \subseteq E'$, so G'_M is indeed a subgraph of G'. Every vertex in G'_M inherits its preference order from G'.

▶ **Lemma 8.** *Let M be any perfect matching in G. Then M is a popular perfect matching in G if and only if M' is a popular perfect matching in the marriage instance G'_M.*

Proof. It follows from [23, Theorem 2] that a perfect matching M' in a marriage instance G'_M is a popular perfect matching if and only if the following condition is satisfied:

- There is no alternating cycle C with respect to M' such that $\mathsf{wt}_{M'}(C) > 0$,

where the edge weight function $\mathsf{wt}_{M'}$ takes values in $\{0, \pm 2\}$ and is analogous to the edge weight function wt_M defined at the start of Section 2.

Direction "⇒". We will show that if M is a popular perfect matching in G then the above condition is satisfied for its canonical realization M' in the marriage instance G'_M. Suppose not. Let C be an alternating cycle with respect to M' in G'_M such that $\mathsf{wt}_{M'}(C) > 0$. Let $N' = M' \oplus C$, i.e., N' is the symmetric difference between M' and C. We have $\mathsf{wt}_{M'}(N') = \mathsf{wt}_{M'}(C) > 0$. By identifying all the clones of the same hospital, the matching N' in G'_M becomes a perfect matching N in G.

For any hospital h with no clone in C, the two sets $M(h)$ and $N(h)$ are the same. For each hospital h with one or more clones in C, the cycle C defines a bijection between $M(h) \setminus N(h)$ and $N(h) \setminus M(h)$. Since $\mathsf{wt}_{M'}(C) > 0$, this means for each h in C, there is a way of comparing elements in $M(h) \setminus N(h)$ with those in $N(h) \setminus M(h)$ such that summed over all vertices in C, the votes in favor of N outnumber the votes in favor of M. Thus, adding up the votes of all vertices in C for M versus N (as per the bijection defined by C), the total number of votes for M is less than the total number of votes for N. Hence it follows from the definition of the function vote (see Eq.(1)) that $\sum_{v \in C} \mathsf{vote}_v(M, N) < 0$. So $\Delta(M, N) < 0$, which contradicts the fact that M is a popular perfect matching in G.

Direction "⇐". Let M' be a popular perfect matching in G'_M; it follows from the characterization of popular perfect matchings in a marriage instance (given above) that there is no alternating cycle C with respect to M' in G'_M such that $\mathsf{wt}_{M'}(C) > 0$. Let N be any perfect matching in G. A realization N' of N in G'_M can be obtained as follows.

- For every edge $(r, h) \in N \cap M$: the edge (r, h_i) is in N' where $(r, h_i) \in M'$.
- For every $(r, h) \in N \setminus M$: in the evaluation of $\mathsf{vote}_h(M, N)$, while comparing the set $M(h) \setminus N(h)$ with the set $N(h) \setminus M(h)$, let s be the resident that h compares r with. So the matching M' contains the edge (s, h_j) for some $j \in \{1, \ldots, \mathsf{cap}(h)\}$; the edge (r, h_j) will be included in N'.

Observe that $\sum_{v \in R \cup H} \mathsf{vote}_v(M, N) = -\sum_C \mathsf{wt}_{M'}(C)$, where the sum is over all alternating cycles C in $M' \oplus N'$. For each alternating cycle $C \in M' \oplus N'$, we have $\mathsf{wt}_{M'}(C) \le 0$, thus $\sum_{v \in R \cup H} \mathsf{vote}_v(M, N) \ge 0$, in other words, $\Delta(M, N) \ge 0$. Since this holds for any perfect matching N, it follows that M is a popular perfect matching in G. ◀

3.2 Constructing a helpful dual certificate

We know that for a perfect matching M to be a popular perfect matching in G, it is not necessary that M' is a popular perfect matching in the marriage instance G'. It suffices for the matching M' to be a popular perfect matching in the subgraph G'_M of G' (by Lemma 8). Observe that being a popular perfect matching in the subgraph G'_M is a more relaxed condition than being a popular perfect matching in G' since the *edge covering constraints* in Lemma 6, i.e., the constraints in property 3, have to be satisfied only for the edges in G'_M rather than all the edges in G' (recall that $E' \supseteq E'_M$).

Recall that there is a function $\mathsf{cost} : E \to \mathbb{R}$ and our goal is to find a *min-cost* popular perfect matching in G. Though we know how to find a min-cost popular perfect matching N in a marriage instance, we do not know in which marriage instance we should run this algorithm (since G'_N depends on the matching N that we seek).

Let M be any popular perfect matching in G. Since the canonical realization M' of M is a popular perfect matching in $G'_M = (R \cup H', E'_M)$, there is a vector $\vec{\gamma} \in \mathbb{R}^n$ (see Lemma 6) that satisfies the following properties where $|R \cup H'| = n$ and $|R| = n_0$.

1. $\gamma_r \in \{0, -2, -4, \ldots, -2(n_0 - 1)\}$ for all $r \in R$.
2. $\gamma_{h_i} \in \{0, 2, 4, \ldots, 2(n_0 - 1)\}$ for all $h_i \in H'$.
3. $\gamma_r + \gamma_{h_i} \geq \mathsf{wt}_{M'}(r, h_i)$ for all $(r, h_i) \in E'_M$.
4. $\sum_{v \in R \cup H'} \gamma_v = 0$.

Let $\vec{\gamma}$ be a dual certificate for M' that minimizes the sum $\sum_{h_i \in H'} \gamma_{h_i}$. The following lemma will be very useful to us.

▶ **Lemma 9.** *For any two clones h_i and h_j of the same hospital h, we have $\gamma_{h_i} \leq \gamma_{h_j} + 2$.*

Proof. Observe that except for their partners in M', the neighborhoods in G'_M of the two clones h_i and h_j of h are identical. Consider any $(r, h_i) \in E'_M$ such that $(r, h_i) \notin M'$. So $(r, h'_t) \in M'$ for some $h' \neq h$. We have:

$$\begin{aligned}
\mathsf{wt}_{M'}(r, h_i) &\leq \mathsf{vote}_r(h_i, M'(r)) + \mathsf{vote}_h(r, M'(h_i)) && (2)\\
&= \mathsf{vote}_r(h_j, M'(r)) + \mathsf{vote}_h(r, M'(h_i)) && (3)\\
&\leq \mathsf{vote}_r(h_j, M'(r)) + \mathsf{vote}_h(r, M'(h_j)) + 2 && (4)\\
&= \mathsf{wt}_{M'}(r, h_j) + 2 && (5)\\
&\leq \gamma_r + \gamma_{h_j} + 2. && (6)
\end{aligned}$$

In the third constraint, $\mathsf{vote}_h(r, M'(h_i)) \leq \mathsf{vote}_h(r, M'(h_j)) + 2$ since $\mathsf{vote}_h(r, M'(h_i)) \leq 1$ and $\mathsf{vote}_h(r, M'(h_j)) \geq -1$.

Suppose $\gamma_{h_i} > \gamma_{h_j} + 2$. Let $(s, h_i) \in M'$. Since M' and $\vec{\gamma}$ are optimal solutions to LP1 and LP2 respectively, we have $\gamma_s + \gamma_{h_i} = \mathsf{wt}_{M'}(s, h_i) = 0$ by complementary slackness. Let us update $\vec{\gamma}$ to $\vec{\gamma}'$ as follows:

- $\gamma'_{h_i} = \gamma_{h_j} + 2$ and $\gamma'_s = -\gamma'_{h_i}$.
- $\gamma'_v = \gamma_v$ for all other vertices v in $R \cup H'$.

Observe that $\sum_{v \in R \cup H'} \gamma'_v = 0$. Thus $\vec{\gamma}'$ satisfies property 4 given in Lemma 6.

Constraints (2)-(6) tell us that $\gamma'_r + \gamma'_{h_i} = \gamma_r + \gamma_{h_j} + 2 \geq \mathsf{wt}_{M'}(r, h_i)$ for any $r \neq s$. Since $\gamma'_s + \gamma'_{h_i} = 0 = \mathsf{wt}_{M'}(s, h_i)$, we have $\gamma'_r + \gamma'_{h_i} \geq \mathsf{wt}_{M'}(r, h_i)$ for all $r \in R$. Thus all edges in E'_M incident to h_i are covered by $\vec{\gamma}'$.

Since $\gamma'_{h_i} = \gamma_{h_j} + 2 < \gamma_{h_i}$, our update ensures that $\gamma'_s \geq \gamma_s$. Because $\gamma'_v = \gamma_v$ for all vertices v other than s and h_i, we have $\gamma'_p + \gamma'_{q_j} \geq \mathsf{wt}_{M'}(p, q_j)$ for every (p, q_j) in E'_M. Thus $\vec{\gamma}'$ satisfies property 3 given in Lemma 6.

So properties 1-4 given in Lemma 6 are satisfied by $\vec{\gamma}'$. Thus $\vec{\gamma}'$ is a dual certificate for M' in G'_M. Moreover, $\sum_{h_i \in H'} \gamma'_{h_i} < \sum_{h_i \in H'} \gamma_{h_i}$, contradicting the choice of $\vec{\gamma}$ as a dual certificate for M' that minimizes this sum. Hence it has to be the case that $\gamma_{h_i} \leq \gamma_{h_j} + 2$. ◀

If $\gamma_{h_i} = 2\ell$ then we will say h_i is in level ℓ in $\vec{\gamma}$. Lemma 9 tells us that all of $h_1, \ldots, h_{\mathsf{cap}(h)}$ are either in the same level (say, ℓ) or in two successive levels (say, ℓ and $\ell+1$) in $\vec{\gamma}$. Let $r_1, \ldots, r_k$ (where $\mathsf{cap}(h) = k$) be the partners of h in M where $r_1 \succ_h \cdots \succ_h r_k$.

▶ **Lemma 10.** *Suppose k' clones of h are in level ℓ and $k - k'$ clones of h are in level $\ell + 1$. Then the k' clones of h in level ℓ have to be matched in M' to $r_1, \ldots, r_{k'}$. That is, $r_1, \ldots, r_{k'}$ are in level ℓ and $r_{k'+1}, \ldots, r_k$ are in level $\ell + 1$ in $\vec{\gamma}$.*

Proof. Suppose at least one of $r_1, \ldots, r_{k'}$ is not in level ℓ. Then r_i, for some $i \leq k'$, is in level $\ell + 1$ and r_j, for some $j > k'$, is in level ℓ. So h_i is in level $\ell + 1$ while h_j is in level ℓ.

Observe that for any neighbor s of h in G such that $s \notin M(h)$, we have $\mathsf{wt}_{M'}(s, h_i) \leq \mathsf{wt}_{M'}(s, h_j)$. This is because $\mathsf{vote}_s(h_i, M) = \mathsf{vote}_s(h_j, M)$ while $\mathsf{vote}_h(s, r_i) \leq \mathsf{vote}_h(s, r_j)$ since h prefers r_i to r_j. Since we have $\gamma_s + \gamma_{h_j} \geq \mathsf{wt}_{M'}(s, h_j) \geq \mathsf{wt}_{M'}(s, h_i)$, let us update $\gamma'_{h_i} = 2\ell$ and $\gamma'_{r_i} = -2\ell$ (so $\gamma'_{r_i} > \gamma_{r_i}$). For any vertex v other than r_i and h_i, let $\gamma'_v = \gamma_v$.

It is easy to see that $\vec{\gamma}'$ is a dual certificate for M' in G'_M and $\sum_{h_i \in H'} \gamma'_{h_i} < \sum_{h_i \in H'} \gamma_{h_i}$, contradicting the choice of $\vec{\gamma}$ as a dual certificate for M that minimizes this sum. Hence it has to be the case that $r_1, \ldots, r_{k'}$ are in level ℓ and $r_{k'+1}, \ldots, r_k$ are in level $\ell + 1$. ◀

We are now ready to prove the main technical lemma in this section. For any $h \in H$, let $r_1, \ldots, r_k$ be all the partners of h in M (so $k = \mathsf{cap}(h)$) and $r_1 \succ_h \cdots \succ_h r_k$.

▶ **Lemma 11.** *Let M be any popular perfect matching in G. For each $h \in H$, there exists an appropriate permutation π_h on $\{1, \ldots, k\}$ (where $k = \mathsf{cap}(h)$) such that $M'' = \cup_{h \in H} \{(r_i, h_{\pi_h(i)}) : 1 \leq i \leq k\}$ is a popular perfect matching in G'.*

Proof. Let $h \in H$ and let $\mathsf{cap}(h) = k$. Consider the following two cases.

Case 1. All the clones of h are in the same level $\ell \in \{0, 1, \ldots, n_0 - 1\}$ in $\vec{\gamma}$.

This is the easy case – we will set $\pi_h : \{1, \ldots, k\} \to \{1, \ldots, k\}$ to be the identity function. So $(r_1, h_1), \ldots, (r_k, h_k)$ are in M''. For all $1 \leq i \leq k$: $\gamma_{r_i} = -2\ell$ and $\gamma_{h_i} = 2\ell$.

Case 2. All the clones of h are *not* in the same level.

So by Lemma 9, all the clones $h_1, \ldots, h_k$ are in two successive levels in $\vec{\gamma}$. We are now ready to define the permutation π_h or equivalently, the partners of $h_1, \ldots, h_k$ in the matching M''. The vertices $r_1, \ldots, r_k$ will be matched in M'' to $h_1, \ldots, h_k$ as follows:

- for $i \in \{1, \ldots, k'\}$, the vertex r_i is matched to $h_{k-k'+i}$;
- for $i \in \{k'+1, \ldots, k\}$, the vertex r_i is matched to $h_{i-k'}$.

In more detail, we know from Lemma 10 that $r_1, \ldots, r_{k'}$ are in level ℓ and $r_{k'+1}, \ldots, r_k$ are in level $\ell+1$. The residents $r_1, \ldots, r_{k'}$ are matched in M'' to $h_{k-k'+1}, \ldots, h_k$, respectively and the residents $r_{k'+1}, \ldots, r_k$ in level $\ell+1$ are matched in M'' to $h_1, \ldots, h_{k-k'}$, respectively. Thus the matching M'' rearranges the clones of h so that the following holds:

- $h_1, \ldots, h_{k-k'}$ are placed in level $\ell + 1$.
- $h_{k-k'+1}, \ldots, h_k$ are placed in level ℓ.

Observe that $M'' = \cup_{h \in H} \{(r_i, h_{\pi_h(i)}) : 1 \leq i \leq k\}$ where in case 1 (i.e., all the clones of h are in the same level), $\pi_h(i) = i$ for all $i \in \{1, \ldots, k\}$ and for case 2, π_h is defined as follows:

$$\pi_h(i) = \begin{cases} k - k' + i & \text{if } 1 \leq i \leq k'; \\ i - k' & \text{if } k' + 1 \leq i \leq k. \end{cases}$$

The entire matching M'' is thus defined. We claim that M'' is the realization of M that we seek.

▷ Claim 12. M'' is a popular perfect matching in G'.

We will prove Claim 12 below. This finishes the proof of Lemma 11. ◀

Proof of Claim 12. We will prove M'' to be a popular perfect matching in G' by defining a dual certificate $\vec{\alpha}$ for M'' in G'. Recall that $\vec{\gamma}$ is a dual certificate for the canonical image M' to be a popular perfect matching in G'_M.

1. For any $r \in R$: set $\alpha_r = \gamma_r$.
2. For any $h \in H$ and $i \in \{1, \ldots, k\}$: set $\alpha_{h_i} = 2t$ where t is h_i's *level* as defined in the proof of Lemma 11.

So in case 1 of the proof of Lemma 11, when all the clones of h are in the same level ℓ, we have $\alpha_{h_i} = 2\ell$ for every $i \in \{1, \ldots, \mathsf{cap}(h)\}$. In case 2 of the proof of Lemma 11, the vertices $h_1, \ldots, h_{k-k'}$ are in level $\ell + 1$ and the vertices $h_{k-k'+1}, \ldots, h_k$ are in level ℓ where $k = \mathsf{cap}(h)$; so $\alpha_{h_i} = 2(\ell + 1)$ for $1 \leq i \leq k - k'$ and $\alpha_{h_i} = 2\ell$ for $k - k' + 1 \leq i \leq k$.

Let $h \in H$. We know the edges $(r_1, h_1), \ldots, (r_k, h_k)$ are in M'. We will now show that $\alpha_{r_i} + \alpha_{h_j} = \mathsf{wt}_{M''}(r_i, h_j)$ for all $i, j \in \{1, \ldots, k\}$.

This is immediate to check for case 1 in the proof of Lemma 11 since $\alpha_{r_i} = -2\ell$ and $\alpha_{h_j} = 2\ell$ for all $i, j \in \{1, \ldots, k\}$. Recall that $r_1 \succ_h \cdots \succ_h r_k$ and $h_1 \succ_r \cdots \succ_r h_k$ for all r adjacent to h in G. So we have $\alpha_{r_i} + \alpha_{h_j} = 0 = \mathsf{wt}_{M''}(r_i, h_j)$ for all $i, j \in \{1, \ldots, k\}$.

We will now check the equality $\alpha_{r_i} + \alpha_{h_j} = \mathsf{wt}_{M''}(r_i, h_j)$ for case 2 in the proof of Lemma 11, so $h_1, \ldots, h_{k-k'}$ are in level $\ell + 1$ and $h_{k-k'+1}, \ldots, h_k$ are in level ℓ.

- For $i \in \{1, \ldots, k'\}$ and $j \in \{k - k' + 1, \ldots, k\}$, observe that $\mathsf{wt}_{M''}(r_i, h_j) = 0$. Since $\alpha_{r_i} = -2\ell$ and $\alpha_{h_j} = 2\ell$, we have $\alpha_{r_i} + \alpha_{h_j} = 0 = \mathsf{wt}_{M''}(r_i, h_j)$ for all these pairs (i, j).
- Similarly, for $i \in \{k' + 1, \ldots, k\}$ and $j \in \{1, \ldots, k - k'\}$, we have $\mathsf{wt}_{M''}(r_i, h_j) = 0$. Since $\alpha_{r_i} = -2(\ell + 1)$ and $\alpha_{h_j} = 2(\ell + 1)$, we have $\alpha_{r_i} + \alpha_{h_j} = 0 = \mathsf{wt}_{M''}(r_i, h_j)$ for all these pairs (i, j).
- Let us now consider a pair (r_i, h_j) where $i \in \{1, \ldots, k'\}$ and $j \in \{1, \ldots, k - k'\}$. We have $\mathsf{wt}_{M''}(r_i, h_j) = 2$ for such an (i, j). Since $\alpha_{r_i} = -2\ell$ and $\alpha_{h_j} = 2(\ell + 1)$, we have $\alpha_{r_i} + \alpha_{h_j} = 2 = \mathsf{wt}_{M''}(r_i, h_j)$ for all these pairs (i, j).
- Finally, let us consider a pair (r_i, h_j) where $i \in \{k' + 1, \ldots, k\}$ and $j \in \{k - k' + 1, \ldots, k\}$. We have $\mathsf{wt}_{M''}(r_i, h_j) = -2$ for such an (i, j). Since $\alpha_{r_i} = -2(\ell + 1)$ and $\alpha_{h_j} = 2\ell$, we have $\alpha_{r_i} + \alpha_{h_j} = -2 = \mathsf{wt}_{M''}(r_i, h_j)$ for all these pairs (i, j).

Recall that the difference between the graphs G'_M and G' is that for every $h \in H$, the edges (r_i, h_j) for $i \neq j$ are missing in G'_M, where the edges $(r_1, h_1), \ldots, (r_k, h_k)$ are in M'. And we just have checked all the edges (r_i, h_j) for all $i, j \in \{1, \ldots, k\}$ are covered by $\vec{\alpha}$.

We will now show that all the *non-matching* edges are covered by $\vec{\alpha}$ as well. That is, for any $(r', h) \in E \setminus M$, we need to show that $\alpha_{r'} + \alpha_{h_j} \geq \mathsf{wt}_{M''}(r', h_j)$ for all $j \in \{1, \ldots, k\}$.

For any $j \in \{1, \ldots, k\}$, let $j = \pi_h(i)$. Observe that $\mathsf{wt}_{M'}(r', h_i) = \mathsf{wt}_{M''}(r', h_j)$. This is because $M'(h_i) = r_i = M''(h_j)$ and $\mathsf{vote}_{r'}(M'', h_j) = \mathsf{vote}_{r'}(M', h_i)$ since r' is matched in both M' and M'' to clones of the same hospital h'.

We have $\alpha_{h_j} = -\alpha_{r_i} = -\gamma_{r_i} = \gamma_{h_i}$ since $(r_i, h_j) \in M''$. So $\gamma_{r'} + \gamma_{h_i} \geq \mathsf{wt}_{M'}(r, h_i)$ implies that $\alpha_{r'} + \alpha_{h_j} \geq \mathsf{wt}_{M''}(r', h_j)$. Thus every non-matching edge (r', h_j) is also covered by $\vec{\alpha}$.

Since $\alpha_s + \alpha_{h_i} = 0$ for any edge $(s, h_i) \in M''$, we have $\sum_{v \in R \cup H'} \alpha_v = 0$. Hence properties 1-4 for dual certificates are satisfied by $\vec{\alpha}$; thus $\vec{\alpha}$ is a dual certificate for M'' in G'. In other words, M'' is a popular perfect matching in G' (by Lemma 6). ◁

Our algorithm. It is now straightforward to show a polynomial time algorithm to find a min-cost popular perfect matching in a hospitals/residents instance $G = (R \cup H, E)$.

1. Compute the corresponding marriage instance $G' = (R \cup H', E')$.
2. Find a min-cost popular perfect matching N' in G' using the algorithm in [23] for marriage instances.
3. Return the corresponding matching N in G by identifying all clones of the same hospital.

For any popular perfect matching M in G, we know there is some realization M'' such that M'' is a popular perfect matching in G' (by Lemma 11); also $\mathsf{cost}(M) = \mathsf{cost}(M'')$. Thus solving the min-cost popular perfect matching problem in the marriage instance G' solves the min-cost popular perfect matching problem in the hospitals/residents instance G. Hence the matching N returned by the above algorithm is a min-cost popular perfect matching in G.

Thus we can conclude Theorem 3 stated in Section 1. We restate this theorem below.

▶ **Theorem 3.** *Let $G = (R \cup H, E)$ be a hospitals/residents instance with $\mathsf{cost} : E \to \mathbb{R}$, where hospitals have capacities and every vertex has a strict ranking of its neighbors. If G admits a perfect matching then a min-cost popular perfect matching in G can be computed in polynomial time.*

4 The many-to-many setting

In this section $G = (R \cup H, E)$ is a many-to-many matching instance. So every vertex $v \in R \cup H$ has a capacity $\mathsf{cap}(v) \geq 1$ associated with it.

Let M be any popular perfect matching in G. Our goal is to show that M can be realized as a popular perfect matching in the marriage instance $G' = (R' \cup H', E')$ where each vertex v in G is replaced by $\mathsf{cap}(v)$ many clones $v_1, \ldots, v_{\mathsf{cap}(v)}$ and every edge $(r, h) \in E$ is replaced by $\mathsf{cap}(r) \cdot \mathsf{cap}(h)$ many edges (r_i, h_j) in E' where $1 \leq i \leq \mathsf{cap}(r)$ and $1 \leq j \leq \mathsf{cap}(h)$.

Each vertex in G' has capacity 1 and it has a strict preference order over its neighbors as described below.

- For $v \in R \cup H$ and $i \in \{1, \ldots, \mathsf{cap}(v)\}$: the preference order of v_i in G' is the same as v's preference order in G where every neighbor u in G gets replaced by all its clones in the order $u_1 \succ \cdots \succ u_{\mathsf{cap}(u)}$.

As done in Section 3, let us first obtain a one-to-one matching M' from M as follows.

- For each edge $(r, h) \in M$, we will choose an index $i \in \{1, \ldots, \mathsf{cap}(r)\}$ and an index $j \in \{1, \ldots, \mathsf{cap}(h)\}$ such that the indices i and j have not been chosen so far and include the edge (r_i, h_j) in M'. Thus M' is a perfect matching in the marriage instance G'.

In fact, as done in Section 3, we could define the matching M' more carefully so that for every $v \in R \cup H$, we have $M'(v_1) \succ_{v_i} \cdots \succ_{v_i} M'(v_{\mathsf{cap}(v)})$ where v_i is any clone of v. As we know, the matching M' need not be a popular perfect matching in G'. Hence let us consider the subgraph $G'_M = (R' \cup H', E'_M)$ of G' whose edge set is as follows:

$$E'_M = M' \cup \{(r_i, h_j) : (r, h) \in E \setminus M \text{ and } 1 \leq i \leq \mathsf{cap}(r),\ 1 \leq j \leq \mathsf{cap}(h)\}.$$

For any $(r, h) \notin M$, all its $\mathsf{cap}(r) \cdot \mathsf{cap}(h)$ many copies $(r_1, h_1), \ldots, (r_{\mathsf{cap}(r)}, h_{\mathsf{cap}(h)})$ are in E'_M while for each edge $(s, h) \in M$, exactly *one* edge, which is the edge in M' – say, (s_i, h_j) – is in E'_M. Thus $E'_M \subseteq E'$, so G'_M is a subgraph of G'.

▶ **Lemma 13.** *Let M be any perfect matching in G. Then M is a popular perfect matching in G if and only if M' is a popular perfect matching in the marriage instance G'_M.*

The proof of the above lemma is analogous to the proof of Lemma 8. Since M' is a popular perfect matching in G'_M, it admits a vector $\vec{\gamma} \in \mathbb{R}^n$ (see Lemma 6) that satisfies the following properties where $|R' \cup H'| = n$ and $|R'| = n_0$:

1. $\gamma_{r_i} \in \{0, -2, -4, \ldots, -2(n_0 - 1)\}$ for all $r_i \in R'$.
2. $\gamma_{h_j} \in \{0, 2, 4, \ldots, 2(n_0 - 1)\}$ for all $h_j \in H'$.
3. $\gamma_{r_i} + \gamma_{h_j} \geq \mathsf{wt}_{M'}(r_i, h_j)$ for all $(r_i, h_j) \in E'_M$.
4. $\sum_{v \in R' \cup H'} \gamma_v = 0$.

Let $\vec{\gamma}$ be a dual certificate for M' that minimizes the value of $\sum_{h_i \in H'} \gamma_{h_i}$. The proofs of Lemmas 14–16 are the same as the proofs of Lemma 9–11, respectively.

▶ **Lemma 14.** *For any two clones h_i and h_j of the same hospital h, we have $\gamma_{h_i} \leq \gamma_{h_j} + 2$.*

If $\gamma_{h_i} = 2\ell$ then we will say h_i is in level ℓ in $\vec{\gamma}$. Lemma 14 tells us that all of $h_1, \ldots, h_{\mathsf{cap}(h)}$ are either in the same level (say, ℓ) or in two successive levels (say, ℓ and $\ell + 1$) in $\vec{\gamma}$. Let $r^1_{i_1}, \ldots, r^k_{i_k}$ (where $k = \mathsf{cap}(h)$) be the partners of h in M where $r^1 \succ_h \cdots \succ_h r^k$.

▶ **Lemma 15.** *Suppose k' clones of h are in level ℓ and $k - k'$ clones of h are in level $\ell + 1$. Then the k' clones of h in level ℓ have to be matched in M' to $r^1_{i_1}, \ldots, r^{k'}_{i_{k'}}$. That is, $r^1_{i_1}, \ldots, r^{k'}_{i_{k'}}$ are in level ℓ and $r^{k'+1}_{i_{k'+1}}, \ldots, r^k_{i_k}$ are in level $\ell + 1$ in $\vec{\gamma}$.*

Let us now define a subgraph G''_M of G'. The edge set E''_M of G''_M has $\mathsf{cap}(h)$ many copies $(r_i, h_{j'})$ of each edge (r_i, h_j) in M', where $1 \leq j' \leq \mathsf{cap}(h)$, and it has all the $\mathsf{cap}(r) \cdot \mathsf{cap}(h)$ copies of each $(r, h) \in E \setminus M$.

Thus $E''_M = E'_M \cup \{(r_i, h_{j'}) : (r_i, h_j) \in M' \text{ and } 1 \leq j' \leq \mathsf{cap}(h)\}$.

▶ **Lemma 16.** *Let M be any popular perfect matching in G. For each $h \in H$, there exists an appropriate permutation π_h on $\{1, \ldots, k\}$ (where $k = \mathsf{cap}(h)$) such that $M'' = \cup_{h \in H}\{(r^j_{i_j}, h_{\pi_h(j)}) : 1 \leq j \leq k\}$ is a popular perfect matching in G''_M where $r^1_{i_1}, \ldots, r^k_{i_k}$ are h's partners in M.*

Thus Lemma 16 tells us that M has a realization as a popular perfect matching M'' in the marriage instance G''_M. However this marriage instance G''_M is not the desired marriage instance G' since each edge $(r_i, h_j) \in M'$ has only $\mathsf{cap}(h)$ many copies $(r_i, h_{j'})$ in G''_M, where $1 \leq j' \leq \mathsf{cap}(h)$. Note that the marriage instance G' includes *all* the $\mathsf{cap}(r) \cdot \mathsf{cap}(h)$ many copies $(r_{i'}, h_{j'})$ of each edge (r, h) in G, where $1 \leq i' \leq \mathsf{cap}(r)$ and $1 \leq j' \leq \mathsf{cap}(h)$.

A useful hospitals/residents instance. For every resident $r \in R$, let us now identify all its clones $r_1, \ldots, r_{\mathsf{cap}(r)}$ in G''_M. Let $\widetilde{G}$ denote the resulting graph.

It will be convenient to swap the two sides of $\widetilde{G}$ so that $\widetilde{G}$ becomes a *many-to-one* matching instance. Thus $\widetilde{G} = (H' \cup R, \widetilde{E})$ where the clones of all hospitals are on the left and residents are on the right. Each vertex $h_i \in H'$ on the left has capacity 1 and each vertex $r \in R$ on the right has $\mathsf{cap}(r) \geq 1$.

The matching M'' in the marriage instance G''_M (see Lemma 16) becomes a many-to-one matching $\widetilde{M}$ in $\widetilde{G}$. We know from Lemma 16 that M'' is a popular perfect matching in G''_M, hence $\widetilde{M}$ is a popular perfect matching in $\widetilde{G}$.

By Lemma 11, the many-to-one popular perfect matching $\widetilde{M}$ in $\widetilde{G}$ has a realization M^* as a popular perfect matching in the corresponding marriage instance. Observe that for every edge (r, h) in G, all the edges (h_i, r) are in $\widetilde{G}$, where $1 \leq i \leq \mathsf{cap}(h)$. So the marriage instance corresponding to $\widetilde{G}$ is $G' = (H' \cup R', E')$ which includes for every edge (r, h) in G, all its $\mathsf{cap}(h) \cdot \mathsf{cap}(r)$ copies (h_i, r_j), where $1 \leq i \leq \mathsf{cap}(h)$ and $1 \leq j \leq \mathsf{cap}(r)$.

Thus it follows that M has a realization M^* as a popular perfect matching in the marriage instance $G' = (R' \cup H', E')$. So we can conclude the following lemma.

▶ **Lemma 17.** *Let M be a popular perfect matching in a many-to-many matching instance $G = (R \cup H, E)$ where vertices have capacities and strict preferences. Then M has a realization M^* as a popular perfect matching in the corresponding marriage instance $G' = (R' \cup H', E')$.*

Hence solving the min-cost popular perfect matching problem in the marriage instance G' solves the min-cost popular perfect matching problem in the many-to-many matching instance G. Since $\mathsf{cap}(r) \leq |H|$ for every $r \in R$ and $\mathsf{cap}(h) \leq |R|$ for every $h \in H$, the number of vertices in G' is $O(|R| \cdot |H|)$. Thus the size of G' is polynomial in the size of G.

As we know, a min-cost popular perfect matching in G' can be computed in polynomial time [23]. Thus Theorem 5 stated in Section 1 follows. We restate this theorem below.

▶ **Theorem 5.** *Let $G = (R \cup H, E)$ be a many-to-many matching instance with $\mathsf{cost} : E \to \mathbb{R}$, where vertices have capacities and every vertex has a strict ranking of its neighbors. If G admits a perfect matching then a min-cost popular perfect matching in G can be computed in polynomial time.*

Concluding remarks. Given a hospitals/residents instance $G = (R \cup H, E)$ where each vertex has a strict preference order over its neighbors and every edge has an associated cost, we showed that if G admits a perfect matching, then a min-cost popular perfect matching in G can be computed in polynomial time. This result generalizes to the many-to-many setting.

Our method of reducing the min-cost popular perfect matching problem in a hospitals/residents instance G to the min-cost popular perfect matching problem in the corresponding marriage instance G' does not work for the min-cost popular *maximum* matching problem. The computational complexity of the min-cost popular maximum matching problem in a hospitals/residents instance is open.

References

1. A. Abdulkadiroğlu and T. Sönmez. School choice: a mechanism design approach. *American Economic Review*, 93(3):729–747, 2003.
2. D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.
3. G. Askalidis, N. Immorlica, A. Kwanashie, D. Manlove, and E. Pountourakis. Socially stable matchings in the hospitals/residents problem. In *Proceedings of the 13th International Symposium on Algorithms and Data Structures, WADS*, pages 85–96, 2013.
4. S. Baswana, P. P. Chakrabarti, S. Chandran, Y. Kanoria, and U. Patange. Centralized admissions for engineering colleges in India. *INFORMS Journal on Applied Analytics*, 49(5):338–354, 2019.
5. P. Biro, R. W. Irving, and D. F. Manlove. Popular matchings in the marriage and roommates problems. In *Proceedings of the 7th International Conference on Algorithms and Complexity, CIAC*, pages 97–108, 2010.
6. P. Biro, D. F. Manlove, and S. Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411:1828–1841, 2010.
7. C. Blair. The lattice structure of the set of stable matchings with multiple partners. *Mathematics of Operations Research*, 13:619–628, 1988.
8. F. Brandl and T. Kavitha. Two problems in max-size popular matchings. *Algorithmica*, 81(7):2738–2764, 2019.
9. Canadian Resident Matching Service. How the matching algorithm works. `http://carms.ca/algorithm.htm`.
10. Á. Cseh. Popular matchings. Trends in Computational Social Choice, Ulle Endriss (ed.), 2017.
11. Á. Cseh, C.-C. Huang, and T. Kavitha. Popular matchings with two-sided preferences and one-sided ties. *SIAM Journal on Discrete Mathematics*, 31(4):2348–2377, 2017.

12 Nicolas de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. L'Imprimerie Royale, 1785.

13 Y. Faenza, T. Kavitha, V. Powers, and X. Zhang. Popular matchings and limits to tractability. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2790–2809, 2019.

14 D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962.

15 D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.

16 P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Science*, 20:166–173, 1975.

17 K. Hamada, K. Iwama, and Shuichi Miyazaki. The hospitals/residents problem with lower quotas. *Algorithmica*, 74(1):440–465, 2016.

18 C.-C. Huang. Classified stable matching. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1235–1253, 2010.

19 C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. *Information and Computation*, 222:180–194, 2013.

20 R. W. Irving, D. F. Manlove, and S. Scott. The hospitals/residents problem with ties. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory, SWAT*, pages 259–271, 2000.

21 R. W. Irving, D. F. Manlove, and S. Scott. Strong stability in the hospitals/residents problem. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, pages 439–450, 2003.

22 T. Kavitha. A size-popularity tradeoff in the stable marriage problem. *SIAM Journal on Computing*, 43(1):52–71, 2014.

23 T. Kavitha. Maximum matchings and popularity. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 85:1–85:21, 2021.

24 S. Merrill and B. Grofman. *A unified theory of voting: directional and proximity spatial models*. Cambridge University Press, 1999.

25 M. Nasre and P. Nimbhorkar. Popular matchings with lower quotas. In *Proceedings of the 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, pages 44:1–44:15, 2017.

26 M. Nasre, P. Nimbhorkar, K. Ranjan, and A. Sarkar. Popular matchings in the hospital-residents problem with two-sided lower quotas. In *Proceedings of the 41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, pages 30:1–30:21, 2021.

27 M. Nasre, P. Nimbhorkar, K. Ranjan, and A. Sarkar. Popular critical matchings in the many-to-many setting, 2022. `arXiv:2206.12394`.

28 M. Nasre and A. Rawat. Popularity in the generalized hospital residents setting. In *Proceedings of the 12th International Computer Science Symposium in Russia, CSR*, pages 245–259, 2017.

29 National Resident Matching Program. Why the Match? `http://www.nrmp.org/whythematch.pdf`.

30 A. E. Roth. Stability and polarization of interest in job matching. *Econometrica*, 53:47–57, 1984.

31 A. E. Roth. On the allocation of residents to rural hospitals: A general property of two-sided matching markets. *Journal of Political Economy*, 54(2):425–427, 1986.

32 U. G. Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54:57–67, 1992.

33 M. Sotomayor. Three remarks on the many-to-many stable matching problem. *Mathematical Social Sciences*, 38:55–70, 1999.

34 C.-P. Teo and J. Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23(4):874–891, 1998.