Computer Science Logic 2017

CSL 2017, August 20–24, 2017, Stockholm, Sweden

Edited by

Valentin Goranko

Mads Dam

LIPIcs – Vol. 82 – CSL 2017 www.dagstuhl.de/lipics

Editors

Valentin Goranko
Department of Philosophy
Stockholm University
Stockholm, Sweden
valentin.goranko@philosophy.su.se

Mads Dam
Department of Theoretical Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden
mfd@kth.se

ACM Classification 1998
A.0 Conference Proceedings, D.2.4 Software/Program Verification, D.3.1 Formal Definitions and Theory, D.3.3 Language Constructs and Features, F. Theory of Computation

ISBN 978-3-95977-045-3

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-045-3.

Publication date
August 2017

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 http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/LIPIcs.CSL.2017.0

ISBN 978-3-95977-045-3 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

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

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

Invited Talks

Contributed Talks

26th EACSL Annual Conference on Compuer Science Logic.
Editors: Valentin Goranko and Mads Dam
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

Computer Science Logic (CSL) is the annual conference of the European Association for Computer Science Logic (EACSL). It is an interdisciplinary conference, spanning across both basic and application oriented research in mathematical logic and computer science. CSL started as a series of international workshops on Computer Science Logic, and became at its sixth meeting the Annual Conference of the EACSL.

The 26th annual EACSL conference Computer Science Logic (CSL 2017) was held in Stockholm from August 20 to August 24, 2017. CSL 2017 was organised jointly by members of the Departments of Philosophy and of Mathematics and Stockholm University, and of the Department of Theoretical Computer Science at KTH Royal Institute of Technology.

CSL 2017 was colocated with the annual European summer meeting of the Association of Symbolic Logic, Logic Colloquium 2017, and the Nordic Logic Summer School of the Scandinavian Logic Society. It also had three affiliated workshops:

- Workshop on Logic and Algorithms in Computational Linguistics, LACompLing 2017,
- Workshop on Logical Aspects of Multi-Agent Systems, LAMAS 2017,
- Workshop on Logic and Automata Theory in memory of Zoltan Esik.

A total of 98 abstracts were registered and 76 of these were followed by full paper submissions to CSL 2017. Each paper was assigned for reviewing to at least three program committee members. The programme committee was assisted by over 100 external reviewers providing additional expertise. The list of external reviewers is included in the proceedings. The program committee worked for two months in total and, after a 4 week long discussion via Easychair, eventually selected 35 papers for presentation at the conference and publication in the proceedings. The overall quality of the submissions was quite good. The number of papers to be accepted was limited by the set overall duration of the conference, though it was also close to the number of submissions considered by the programme committee sufficiently good to be accepted. Still, we had to reject a few good papers. As a new feature for CSL we introduced the option of additional submission of short oral presentations, without publication in the proceedings and subject to lighter refereeing. Eventually, we accepted 8 such short presentations, of which 3 were based on regular submissions which could not be included in the programme as full papers.

As another novel feature, the CSL 2017 conference included a joint special session with Logic Colloquium, featuring two invited highlight speakers from each of the two conferences, in addition to the four plenary invited speakers giving presentations during the regular conference days. The speakers invited by CSL for the joint CSL/LC session were Phokion Kolaitis (University of California Santa Cruz and IBM Research - Almaden) and Wolfgang Thomas (RWTH Aachen University). The invited speakers for the regular CSL program were Laura Kovacs (Technische Universität Wien), Stephan Kreuzer (Technische Universität Berlin), Meena Mahajan (Institute of Mathematical Sciences, Chennai), and Margus Veanes (Microsoft Research). The invited speakers have contributed abstracts that are included in the proceedings.

A special regular item in the CSL programme is the Ackermann Award presentation. This is the EACSL Outstanding Dissertation Award for Logic in Computer Science. This year, the jury decided to give the Ackermann Award for 2017 to Amaury Pouly. The award was officially presented at the conference on August 22, 2017. The citation of the award, an abstract of the thesis and a biographical sketch of the recipient written by Daniel Leivant and Anuj Dawar is included in the proceedings.

26th EACSL Annual Conference on Compuer Science Logic.
Editors: Valentin Goranko and Mads Dam

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

The CSL 2017 conference also saw the presentation of the second Alonzo Church award. This is a joint award of the EACSL, ACM SigLog, EATCS and the Kurt Goedel Society. It is given for an outstanding contribution in the field of Logic and Computation represented by a paper or small group of papers within the past 25 years. This year the award is given jointly to Samson Abramsky, Radha Jagadeesan, Pasquale Malacaria, Martin Hyland, Luke Ong, and Hanno Nickau for providing a fully-abstract semantics for higher-order computation through the introduction of game models, thereby fundamentally revolutionising the field of programming language semantics, and for the applied impact of these models. Their contributions appeared in three papers:

- S. Abramsky, R. Jagadeesan, and P. Malacaria. Full Abstraction for PCF. Information and Computation, Vol. 163, No. 2, pp. 409–470, 2000.
- J. M. E. Hyland and C.-H. Luke Ong. On Full Abstraction for PCF: I, II, and III. Information and Computation, Vol. 163, No. 2, pp. 285–408, 2000.
- H. Nickau. Hereditarily sequential functionals. Proc. Symp. Logical Foundations of Computer Science: Logic at St. Petersburg (eds. A. Nerode and Yu. V. Matiyasevich), Lecture Notes in Computer Science, Vol. 813, pp. 253–264. Springer-Verlag, 1994.

We wish to thank all members of the programme committee and all external reviewers for their hard and highly professional work on reviewing and discussing the papers. Our thanks also go to the members of the organising committee and to the workshops organisers for their valuable contribution to organising CSL 2017. We also wish to thank Anuj Dawar who, as the EACSL president, was always there to help us with opinion and advice, as well as Marc Herbstritt from the Dagstuhl/LIPIcs team for assisting us in the production of the proceedings.

We send our warm thanks to the conference sponsors Stockholm University, including the Departments of Mathematics and Philosophy, KTH Royal Institute of Technology, including the School of Computer Science and Communication and the Department of Theoretical Computer Science, the Swedish Research Council VR, and Prover Technology AB. We also extend our thanks to Stockholm City Council for hosting the conference reception at Stockholm City Hall.

The workshop on Logic and Automata Theory affiliated with CSL 2017 was a tribute to the memory of Zoltan Esik, whom we lost tragically during the last year. Zoltan was actively associated with CSL and the EACSL, organizing CSL 2006 in Szeged, Hungary. All the members of the CSL community wish to express their sadness and sympathy to his family, friends and colleagues.

List of Authors

Benedikt Ahrens
Inria
France
benedikt.ahrens@inria.fr

Dana Angluin
Yale University
New Haven, CT, USA

Timos Antonopoulos
Yale University
New Haven, CT, USA

Bartosz Bednarczyk
University of Wrocław
Poland
bbednarczyk@stud.cs.uni.wroc.pl

Udi Boker
Interdisciplinary Center (IDC)
Herzliya, Israel

Célia Borlido
IRIF, CNRS & Université Paris Diderot - Paris 7
Paris, France

Joseph Boudou
IRIT – Toulouse University
Toulouse, France
joseph.boudou@irit.fr

Matthew de Brecht
Graduate School of Human and Environmental Studies, Kyoto University
Japan
matthew@i.h.kyoto-u.ac.jp

Davide Bresolin
Dept. of Mathematics, University of Padova
Italy
davide.bresolin@unipd.it

Olivier Carton
Université Paris Diderot, LIAFA, UMR 7089
Case 7014, 75 205 Paris Cedex 13, France
Olivier.Carton@liafa.univ-paris-diderot.fr

Witold Charatonik
University of Wrocław
Poland
wch@cs.uni.wroc.pl

Krishnendu Chatterjee
IST Austria
Vienna, Austria

Yijia Chen
School of Computer Science, Fudan University
Shanghai, China
yijiachen@fudan.edu.cn

James Cheney
LFCS, University of Edinburgh
Edinburgh, United Kingdom
jcheney@inf.ed.ac.uk

J.R.B. Cockett
Department of Computer Science, University of Calgary
Calgary, Alberta, CANADA
robin@ucalgary.ca

Silke Czarnetzki
Wilhelm-Schickard Institut, Universität Tübingen
Tübingen, Germany

Martín Diéguez
Institut de Recherche en Informatique de Toulouse, Toulouse University
Toulouse, France

Wolfgang Dvořák
TU Wien and University of Vienna
Vienna, Austria

Martín H. Escardó
School of Computer Science, University of Birmingham
United Kingdom
m.escardo@cs.bham.ac.uk

David Fernández-Duque
Institut de Recherche en Informatique de Toulouse, Toulouse University
Toulouse, France

26th EACSL Annual Conference on Compuer Science Logic.
Editors: Valentin Goranko and Mads Dam
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Olivier Finkel
Equipe de Logique Mathématique
Institut de Mathématiques de Jussieu - Paris Rive Gauche
CNRS et Université Paris 7, France.
finkel@math.univ-paris-diderot.fr

Dana Fisman
Ben-Gurion University
Be'er Sheva, Israel

Jörg Flum
Mathematisches Institut, Universität Freiburg
Germany
joerg.flum@math.uni-freiburg.de

Wan Fokkink
Vrije Universiteit Amsterdam
The Netherlands

Mai Gehrke
IRIF, CNRS & Université Paris Diderot - Paris 7
Paris, France

Ulysse Gérard
Inria Saclay & LIX/École Polytechnique
Palaiseau, France

Dan R. Ghica
University of Birmingham
United Kingdom

Rob van Glabbeek
Data61, CSIRO and University of New South Wales
Sydney, Australia

Maria João Gouveia
CEMAT-CIÊNCIAS
(UID/Multi/04621/2013)
Faculdade de Ciências, Universidade de Lisboa
Portugal
mjgouveia@fc.ul.pt

Berit Grußien
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany
grussien@informatik.hu-berlin.de

Erich Grädel
Mathematical Foundations of Computer Science, RWTH Aachen University
Aachen, Germany
graedel@logic.rwth-aachen.de

Alessio Guglielmi
Department of Computer Science, University of Bath
United Kingdom
a.guglielmi@bath.ac.uk

Miika Hannula
Department of Computer Science, University of Auckland
New Zealand
m.hannula@auckland.ac.nz

Gerco van Heerdt
University College London
United Kingdom
gerco.heerdt@ucl.ac.uk

Monika Henzinger
University of Vienna
Vienna, Austria

Rostislav Horčík
Institute of Computer Science, Czech Academy of Sciences
Czech Republic
horcik@cs.cas.cz

Xuangui Huang
Department of Computer Science, Shanghai Jiao Tong University
China
stslxg@gmail.com

Achim Jung
University of Birmingham
United Kingdom

Emanuel Kieroński
University of Wrocław
Poland
kiero@cs.uni.wroc.pl

Bartek Klin
University of Warsaw
Poland

Cory M. Knapp
School of Computer Science, University of Birmingham
United Kingdom
cmk497@cs.bham.ac.uk

Phokion G. Kolaitis
University of California Santa Cruz and IBM Research - Almaden
United States

Laura Kovács
Vienna University of Technology
Austria
laura.kovacs@tuwien.ac.at

Andreas Krebs
Wilhelm-Schickard Institut, Universität Tübingen
Tübingen, Germany

Stephan Kreutzer
TU Berlin
Germany
stephan.kreutzer@tu-berlin.de

Dominique Lecomte
Projet Analyse Fonctionnelle Institut de Mathématiques de Jussieu - Paris Rive Gauche Université Paris 6, and Université de Picardie, I.U.T. de l'Oise, site de Creil
France
dominique.lecomte@upmc.fr

Mateusz Łełyk
University of Warsaw
Poland

J-S. Lemay
Department of Mathematics and Statistics, University of Calgary
Calgary, Alberta, CANADA
jeansimon.lemay@ucalgary.ca

Veronika Loitzenbauer
University of Vienna and Bar-Ilan University
Vienna, Austria and Ramat Gan, Israel

Aliaume Lopez
ENS Cachan, Université Paris-Saclay
France

Martin Lück
Leibniz Universität Hannover, Institut für Theoretische Informatik
Appelstraße 4, 30167 Hannover, Germany
lueck@thi.uni-hannover.de

Peter LeFanu Lumsdaine
Department of Mathematics, Stockholm University
Sweden
p.l.lumsdaine@math.su.se

Meena Mahajan
The Institute of Mathematical Sciences, HBNI
Chennai, India
meena@imsc.res.in

Dale Miller
Inria Saclay & LIX/École Polytechnique
Palaiseau, France

Tommaso Moraschini
Institute of Computer Science, Czech Academy of Sciences
Czech Republic
moraschini@cs.cas.cz

Emilio Muñoz-Velasco
Dept. of Applied Mathematics, University of Málaga
Spain
ejmunoz@uma.es

Koko Muroya
University of Birmingham
United Kingdom
k.muroya@cs.bham.ac.uk

Mateus de Oliveira Oliveira
Department of Informatics - University of Bergen
Norway
mateus.oliveira@uib.no

Wilmer Ricciotti
LFCS, University of Edinburgh
Edinburgh, United Kingdom
wricciot@inf.ed.ac.uk

Benedikt Pago
RWTH Aachen University
Germany
benedikt.pago@rwth-aachen.de

Wied Pakusa
University of Oxford
England
wied.pakusa@cs.ox.ac.uk

Arno Pauly
Départment d'Informatique, Université libre de Bruxelles
Brussels, Belgium
arno.m.pauly@gmail.com

Alice Pavaux
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030
France

Benjamin Ralph
Department of Computer Science, University of Bath
United Kingdom
b.d.ralph@bath.ac.uk

Matteo Sammartino
University College London
United Kingdom
m.sammartino@ucl.ac.uk

Abhisekh Sankaran
Department of Computer Science and Engineering, Indian Institute of Technology Bombay
India
abhisekh@cse.iitb.ac.in

Luigi Santocanale
Laboratoire d'Informatique Fondamentale de Marseille
UMR 7279, CNRS AMU, France
luigi.santocanale@lif.univ-mrs.fr

Guido Sciavicco
Dept. of Mathematics and Computer Science, University of Ferrara
Italy
guido.sciavicco@unife.it

Alexandra Silva
University College London
United Kingdom
alexandra.silva@ucl.ac.uk

Wolfgang Thomas
RWTH Aachen
Germany
thomas@automata.rwth-aachen.de

Margus Veanes
Microsoft Research
Redmond, USA
margus@microsoft.com

Oleg Verbitsky
Institut für Informatik,
Humboldt-Universität zu Berlin
Unter den Linden 6, D-10099 Berlin,
Germany

Amanda Vidal
Institute of Computer Science, Czech Academy of Sciences
Czech Republic
amanda@cs.cas.cz

Marc de Visme
École Normale Supérieure de Paris
France

Vladimir Voevodsky
Institute for Advanced Study, Princeton
NJ, USA
vladimir@ias.edu

Glynn Winskel
Computer Laboratory, University of Cambridge
United Kingdom

Faried Abu Zaid
Department of Computer Science and Automation, TU Ilmenau
Ilmenau, Germany
Faried.Abu-Zaid@tu-ilmenau.de

Maksim Zhukovskii
Moscow Institute of Physics and Technology,
Department of Discrete Mathematics
Moscow, Russia

Conference Organization

Program Committee

- Parosh Aziz Abdulla (University of Uppsala)
- Lars Birkedal (University of Aarhus)
- Nikolaj Bjorner (Microsoft Research)
- Maria Paola Bonacina (Universitá degli Studi di Verona)
- Patricia Bouyer-Decitre (LSV, CNRS & ENS Cachan)
- Agata Ciabattoni (University of Viena)
- Thierry Coquand (University of Gothenburg)
- Mads Dam (KTH, Stockholm, co-chair)
- Ugo Dal Lago (University of Bologna)
- Anuj Dawar (Cambridge University)
- Valentin Goranko (Stockholm University, co-chair)
- Maribel Fernandez (King's College London)
- Martin Grohe (RWTH Aachen)
- Lauri Hella (University of Tampere)
- Joost-Pieter Katoen (RWTH Aachen)
- Orna Kupferman (University of Jerusalem)
- Leonid Libkin (University of Edinburgh)
- Angelo Montanari (University of Udine)
- Catuscia Palamidessi (Paris, INRIA)
- Frank Pfenning (Carnegie Mellon University, Pittsburgh)
- Ram Ramanujam (Institute of Mathematical Sciences, Chennai)
- Jean-Francois Raskin (University of Bruxelles)
- Thomas Schwentick (University of Dortmund)
- Viorica Sofronie-Stokkermans (University of Koblenz-Landau)
- Thomas Streicher (University of Darmstadt)
- Jean-Marc Talbot (University of Aix-Marseille)
- Luca Viganó (King's College London)
- Ron van der Meyden (UNSW Australia)
- Lijun Zhang (Chinese Academy of Sciences, Beijing)

26th EACSL Annual Conference on Compuer Science Logic.
Editors: Valentin Goranko and Mads Dam

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

Organising Committee

- Stefan Buijsman, Department of Philosophy, Stockholm University
- Mads Dam (co-chair), Department of Theoretical Computer Science, KTH Royal Institute of Technology
- Jacopo Emmenegger, Department of Mathematics, Stockholm University
- Valentin Goranko (co-chair), Department of Philosophy, Stockholm University
- Dilian Gurov (workshops chair), Department of Theoretical Computer Science, KTH Royal Institute of Technology
- Eric Johannesson, Department of Philosophy, Stockholm University
- Vera Koponen, Department of Mathematics, Uppsala University
- Johan Lindberg, Department of Mathematics, Stockholm University
- Roussanka Loukanova, Department of Mathematics, Stockholm University
- Peter LeFanu Lumsdaine, Department of Mathematics, Stockholm University
- Anders Lundstedt, Department of Philosophy, Stockholm University
- Karl Nygren, Department of Philosophy, Stockholm University
- Erik Palmgren (co-chair), Department of Mathematics, Stockholm University
- Thomas Tuerk, Department of Theoretical Computer Science, KTH Royal Institute of Technology

External Reviewers

Toshiyasu Arai
Vincent Aravantinos
Pablo Arrighi
Federico Aschieri
Philippe Balbiani
Pablo Barceló
Andrej Bauer
Dietmar Berwanger
Ales Bizjak
Valentin Blot
Joseph Boudou
Thierry Boy de La Tour
Laura Bozzelli
Julian Bradfield
Sabine Broda
Antonio Bucciarelli
Yu-Fang Chen
Ranald Clouston
Thomas Colcombet
Marco Console
Ioana Cristescu
Giovanna D'Agostino
Nils Anders Danielsson
Anupam Das
Dario Della Monica
Gilles Dowek
Andrej Dudenhefner
Claudia Faggian
Alessandro Farinelli
Francicleber Ferreira
Nathanaël Fijalkow
Dana Fisman
Guido Gherardi
Sujata Ghosh
Stéphane Graham-Lengrand
Erich Grädel
Giulio Guerrieri
Tobias Heindel
Olivier Hermant
Christina Jansen
David N. Jansen
Vincent Juge
Marie Kerjean
Antonina Kolokolova
Denis Kuperberg
James Laird
François Lamarche
Dominique Larchey-Wendling
Olivier Laurent
Massimo Lauria
Pierre Lescanne
Andrzej Lingas
Wanwei Liu
Kamal Lodaya
Markus Lohrey
Etienne Lozes
Kerkko Luosto
Maria Emilia Maietti
Sérgio Marcelino
Christoph Matheja
Damiano Mazza
Paul-Andre Mellies
Samuel Mimram
Kenji Miyamoto
Fabio Mogavero
Alberto Molinari
Mohammad Mousavi
Agata Murawska
Markus Müller-Olm
Martin Otto
Arno Pauly
Guillermo Perez
Ján Pich
Elaine Pimentel
Sophie Pinchinat
Adolfo Piperno
Paolo Pistone
Ian Pratt-Hartmann
Tahiry Rabehaja
Revantha Ramanayake
Joao Rasga
Laurent Regnier
Jakob Rehof
Umberto Rivieccio
Raine Rönnholm
Joshua Sack
Alexis Saurin
Matthias Schroeder
Luc Segoufin
Sunil Easaw Simon
Michał Skrzypczak
Michael Soltys
Bas Spitters
Ludwig Staiger
Sorin Stratulat
S P Suresh
Evgenia Ternovska
Kazushige Terui
Alwen Tiu
Christian Urban
Hans van Ditmarsch
Rob van Glabbeek
Lionel Vaux
Yaron Velner
Uwe Waldmann
Fan Yang
Richard Zach
Thomas Zeume
Stanislav Živný

26th EACSL Annual Conference on Compuer Science Logic.
Editors: Valentin Goranko and Mads Dam
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The Ackermann Award 2017

Anuj Dawar[*1] **and Daniel Leivant**[*2]

1 **University of Cambridge, Cambridge, UK**
2 **Indiana University, Bloomington, IN, USA**

Abstract

The Ackermann Award is the EACSL Outstanding Dissertation Award for Logic in Computer Science. It is presented during the annual conference of the EACSL (CSL'xx). This contribution reports on the 2017 edition of the award.

1998 ACM Subject Classification F.3 Logics and Meanings of Programs, F.4 Mathematical Logic and Formal Languages

Keywords and phrases Ackermann Award, Computer Science, Logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.1

Category Award Description

1 The Ackermann Award 2017

The thirteenth Ackermann Award is presented at CSL'17 in Stockholm, Sweden. The 2017 Ackermann Award was open to any PhD dissertation in the topics represented at the annual CSL and LICS conferences which were formally accepted as theses for the award of a PhD degree at a university or equivalent institution between 1 January 2015 and 31 December 2016. The Jury received fourteen nominations for the Ackermann Award 2017. The candidates came from a number of different countries across the world. The institutions at which the nominees obtained their doctorates represent eleven different countries in Europe and North America.

The topics covered a wide range of Logic and Computer Science as represented by the LICS and CSL Conferences. All submissions were of a very high standard and contained remarkable contributions to their particular fields. The Jury wishes to extend its congratulations to all nominated candidates for their outstanding work. The Jury encourages them to continue their scientific careers and hopes to see more of their work in the future.

The wide range of excellent candidates presented the jury with a difficult task. After an extensive discussion, one candidate stood out and the jury unanimously decided to award the **2017 Ackermann Award** to:

Amaury Pouly from France, for his thesis
Continuous Models of Computation: From Computability to Complexity
approved jointly by the École Polytechnique (France) and the Universidade do Algarve (Portugal), in 2015,
supervised by Olivier Bournez and Daniel Graça.

[*] For the Jury of the EACSL Ackermann Award.

Citation. Amaury Pouly receives the *2017 Ackermann Award* of the European Association of Computer Science Logic (EACSL) for his thesis

Continuous Models of Computation: From Computability to Complexity.

His thesis offers a novel insight into computational complexity of analog devices, leading through deep and difficult landmark results to a strong corroboration of Church-Turing Thesis across digital and analog computation models, and of identifying feasibility with PTime across such models.

Background of the Thesis. Turing gave in 1936 a compelling analysis of digital computation devices, thereby providing intuitive evidence for the Church-Turing Thesis, that is the identification of digital computability with computability by Turing machines. This identification does not rule out, however, the existence of physical analog computation devices that are more powerful than Turing machines.

A mathematical model of analog computing was proposed in 1941 by Claude Shannon, dubbed the General Purpose Analog Computer (GPAC), and inspired by the Differential Analyzer a 1931 physical device built at MIT. Recent renewed interest in analog computing emanates from a general interest in novel models of computation, such as quantum and biological systems, as well as hybrid systems, where continuous computation plays a role.

Nonetheless, GPAC remains a solid paradigm for analog computing, because it is realistically implementable, is entirely continuous for both time and space, and can equivalently be described as the class of dynamical systems defined by differential equations, indeed by differential equations of a particularly simple form. While digital computers have replaced analog computers, the theoretical question of whether analog computers might be more powerful remains all the more pertinent with the advent of other models of computation, such as quantum and biological computers.

In recent years Graça and Bournez showed that GPACs are indeed equivalent to Turing machines: the two models compute the same real-valued functions. A key component of this equivalence is the characterization of GPAC by Costa and Graça (2003) in terms of polynomial differential equations. This corroborates the physical Church-Turing Thesis concerning computability. However, the real-life practicality of that equivalence hinges on the complexity of the mutual simulations. To resolve this issue, we need an appropriate notion of computational complexity for analog computing. Previous attempts to define such measures failed, primarily because time is not an appropriate complexity parameter for continuous computing: indeed, it can be re-scaled at will. The challenge is, therefore, to identify a complexity parameter that corresponds in analog computing to computation time in digital computing. This is the point of departure of Pouly's thesis.

Achievements. Identifying an appropriate complexity measure for GPACs required an important preliminary insight: the notions corresponding to time and to space are independent in continuous systems, as they can be traded for each other via scaling. This contrasts with discrete systems, where time bounds imply space bounds. An adequate complexity measure for GPACs must therefore refer to bounds of both quantities. Pouly's elegant solution was to adopt as a single complexity measure the length of the trajectory (computation) curve. That length is large if the system "blows up", consuming space, OR if it has a rapidly changing trajectory, consuming time. Either case represents computational hardness.

Pouly proceeds to define mutual simulations of GPACs and Turing machines that are polynomial with respect to computational complexity. This corroborates the physical

Church-Turing thesis at the practical level: no continuous physical device yields a super-polynomial speed-up over Turing machines.

The proof of that equivalence represents a highly complex and mature mathematical achievement, combining tools and insights from various disciplines, including classical analysis, numerical methods, dynamical systems theory, program verification, and computational complexity theory.

Extensions and Applications. Certain facets of Pouly's central proof are of substantial independent interest. Already in its initial phase, the proof includes an adaptive Taylor algorithm to solve polynomial initial-value differential equation problems over unbounded intervals, of polynomial computational complexity. Pouly also gives an implicit characterization of PTime for Turing machines in terms of differential equations, a result that earned him a Best Student Award at ICALP 2016. Moreover, the technology developed in the thesis opens the door to viewing differential equations as a computation model and as a general tool for implicit computational complexity. For example, following his thesis Pouly developed (with Bournez) a differential equation that maybe viewed as universal.

Broad Impact. We have been witnessing for some time the emergence of a broad array of computation models, related to various areas of databases, mathematics, physics, and biology. Unfortunately missing from this development are unifying concepts, themes, techniques, and results. Pouly's thesis is a remarkable contribution to the search of such unity. Surprisingly, it also provides new tools for both digital computing (such as implicit complexity via differential equations) and analog computing (efficient algorithms for differential equations).

Biographical Sketch. Amaury Pouly completed his early education in Lyon, France, obtaining a Bachelor's and Master's degree from the École Normale Supérieure de Lyon. His PhD work was jointly carried out in the LIX laboratory of the École Polytechnique (under the supervision of Olivier Bournez) and the University of Algarve in Portugal (under the supervision of Daniel Graça). Since completing his PhD in 2015, he has been working as a postdoctoral research assistant to Joel Ouaknine, first at Oxford and more recently at the Max Planck Institute in Saarbrücken.

2 Jury

The Jury for the **Ackermann Award 2017** consisted of eight members, two of them *ex officio*, namely, the president and the vice-president of EACSL. In addition, the jury also included a representative of SigLog (the ACM Special Interest Group on Logic and Computation).

The members of the jury were:

- Anuj Dawar (University of Cambridge), the president of EACSL,
- Orna Kupferman (Hebrew University of Jerusalem),
- Daniel Leivant (Indiana University, Bloomington),
- Dale Miller (INRIA Saclay), SigLog representative,
- Luke Ong (University of Oxford),
- Jean-Éric Pin (CNRS and Université Paris Diderot),
- Simona Ronchi Della Rocca (University of Torino), the vice-president of EACSL.
- Thomas Schwentick (TU Dortmund University)

3 Previous winners

Previous winners of the Ackermann Award were

2005, Oxford:
Mikołaj Bojańczyk from Poland,
Konstantin Korovin from Russia, and
Nathan Segerlind from the USA.

2006, Szeged:
Balder ten Cate from the Netherlands, and
Stefan Milius from Germany.

2007, Lausanne:
Dietmar Berwanger from Germany and Romania,
Stéphane Lengrand from France, and
Ting Zhang from the People's Republic of China.

2008, Bertinoro:
Krishnendu Chatterjee from India.

2009, Coimbra:
Jakob Nordström from Sweden.

2010, Brno:
no award given.

2011, Bergen:
Benjamin Rossman from USA.

2012, Fontainebleau:
Andrew Polonsky from Ukraine, and
Szymon Toruńczyk from Poland.

2013, Turin:
Matteo Mio from Italy.

2014, Vienna:
Michael Elberfeld from Germany.

2015, Berlin:
Hugo Férée from France, and
Mickael Randour from Belgium

2016, Marseille:
Nicolai Kraus from Germany

Detailed reports on their work appeared in the CSL proceedings and are also available on the EACSL homepage.

Schema Mappings: Structural Properties and Limits

Phokion G. Kolaitis

University of California, Santa Cruz, CA, USA; and
IBM Almaden Research Center, San Jose, CA, USA

Abstract

A schema mapping is a high-level specification of the relationship between two database schemas. For the past fifteen years, schema mappings have played an essential role in the modeling and analysis of important data inter-operability tasks, such as data exchange and data integration. Syntactically, schema mappings are expressed in some schema-mapping language, which, typically, is a fragment of first-order logic or second-order logic. In the first part of the talk, we will introduce the main schema-mapping languages, will discuss the fundamental structural properties of these languages, and will then use these structural properties to obtain characterizations of various schema-mapping languages in the spirit of abstract model theory. In the second part of the talk, we will examine schema mappings from a dynamic viewpoint by considering sequences of schema mappings and studying the convergence properties of such sequences. To this effect, we will introduce a metric space that is based on a natural notion of distance between sets of database instances and will investigate pointwise limits and uniform limits of sequences of schema mappings. Among other findings, it will turn out that the completion of this metric space can be described in terms of graph limits arising from converging sequences of homomorphism densities.

Much of the material presented in this talk is drawn from [1, 2, 3, 4].

1998 ACM Subject Classification H.2.5 [Heterogeneous Databases] Data Translation, D.3.2 [Language Classifications] Constraint and Logic Languages

Keywords and phrases Logic and databases, schema mappings, data exchange, metric spaces

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.2

Category Invited Talk

References

1 Balder ten Cate and Phokion G. Kolaitis. Structural characterizations of schema-mapping languages. In Ronald Fagin, editor, *Database Theory – ICDT 2009, 12th Int'l Conf., St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, pages 63–72. ACM, 2009. doi:10.1145/1514894.1514903.

2 Balder ten Cate and Phokion G. Kolaitis. Structural characterizations of schema-mapping languages. *Commun. ACM*, 53(1):101–110, 2010. doi:10.1145/1629175.1629201.

3 Balder ten Cate and Phokion G. Kolaitis. Schema mappings: A case of logical dynamics in database theory. In Alexandru Baltag and Sonja Smets, editors, *Johan van Benthem on Logic and Information Dynamics*, pages 67–100. Springer, 2014. doi:10.1007/978-3-319-06025-5_3.

4 Phokion G. Kolaitis, Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. Limits of schema mappings. In Wim Martens and Thomas Zeume, editors, *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*, volume 48 of *LIPIcs*, pages 19:1–19:17. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICDT.2016.19.

First-Order Interpolation and Grey Areas of Proofs*

Laura Kovács

Vienna University of Technology, Vienna, Austria
laura.kovacs@tuwien.ac.at

Abstract

Interpolation is an important technique in computer aided verification and static analysis of programs. In particular, interpolants extracted from so-called local proofs are used in invariant generation and bounded model checking. An interpolant extracted from such a proof is a boolean combination of formulas occurring in the proof.

In this talk we first describe a technique of generating and optimizing interpolants based on transformations of what we call the "grey area" of local proofs. Local changes in proofs can change the extracted interpolant. Our method can describe properties of extracted interpolants obtained by such proof changes as a pseudo-boolean constraint. By optimizing solutions of this constraint we also improve the extracted interpolants. Unlike many other interpolation techniques, our technique is very general and applies to arbitrary theories. Our approach is implemented in the theorem prover Vampire and evaluated on a large number of benchmarks coming from first-order theorem proving and bounded model checking using logic with equality, uninterpreted functions and linear integer arithmetic. Our experiments demonstrate the power of the new techniques: for example, it is not unusual that our proof transformation gives more than a tenfold reduction in the size of interpolants.

While local proofs admit efficient interpolation algorithms, standard complete proof systems, such as superposition, for theories having the interpolation property are not necessarily complete for local proofs. In this talk we therefore also investigate interpolant extraction from non-local proofs in the superposition calculus and prove a number of general results about interpolant extraction and complexity of extracted interpolants. In particular, we prove that the number of quantifier alternations in first-order interpolants of formulas without quantifier alternations is unbounded. This result has far-reaching consequences for using local proofs as a foundation for interpolating proof systems - any such proof system should deal with formulas of arbitrary quantifier complexity.

1998 ACM Subject Classification F.3.1. Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic, I.2.3 Deduction and Theorem Proving

Keywords and phrases Theorem proving, interpolation, proof transformations, constraint solving, program analysis

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.3

Category Invited Talk

Acknowledgements. This talk is based on joint work with Andrei Voronkov (University of Manchester, Vienna University of Technology and EasyChair).

* This work was supported by the ERC Starting Grant 2014 SYMCAR 639270, the Wallenberg Academy Fellowship 2014 TheProSE, the Swedish VR grant GenPro D0497701 and the FWF projects S11403-N23 and S11409-N23. We also acknowledge support from the FWF project W1255-N23.

Current Trends and New Perspectives for First-Order Model-Checking*

Stephan Kreutzer

Chair for Logic and Semantics, Technical University Berlin, Berlin, Germany
stephan.kreutzer@tu-berlin.de

Abstract

The model-checking problem for a logic $\mathcal{L}$ is the problem of decidig for a given formula $\varphi \in \mathcal{L}$ and structure $\mathfrak{A}$ whether the formula is true in the structure, i.e. whether $\mathfrak{A} \models \varphi$.

Model-checking for logics such as *First-Order Logic* (FO) or *Monadic Second-Order Logic* (MSO) has been studied intensively in the literature, especially in the context of algorithmic meta-theorems within the framework of parameterized complexity. However, in the past the focus of this line of research was model-checking on classes of *sparse* graphs, e.g. planar graphs, graph classes excluding a minor or classes which are nowhere dense. By now, the complexity of first-order model-checking on sparse classes of graphs is completely understood. Hence, current research now focusses mainly on classes of dense graphs.

In this talk we will briefly review the known results on sparse classes of graphs and explain the complete classification of classes of sparse graphs on which first-order model-checking is tractable. In the second part we will then focus on recent and ongoing research analysing the complexity of first-order model-checking on classes of dense graphs.

1998 ACM Subject Classification F.4.1 Computational Logic

Keywords and phrases Finite Model Theory, Computational Model Theory, Algorithmic Meta Theorems, Model-Checking, Logical Approaches in Graph Theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.4

Category Invited Talk

1 Introduction

The *model checking problem* $\mathrm{MC}(\mathcal{L})$ for a logic $\mathcal{L}$ is the problem to decide, given as input a formula $\varphi \in \mathcal{L}$ and a structure $\mathfrak{A}$, whether φ is true in $\mathfrak{A}$. Often the model checking problem is relativised to a particular class $\mathcal{C}$ of structures. We define $\mathrm{MC}(\mathcal{L}, \mathcal{C})$ as the restriction of $\mathrm{MC}(\mathcal{L})$ to structures in $\mathcal{C}$.

Understanding the model checking complexity for important logical systems such as first-order (FO) and monadic second-order logic (MSO) but also various temporal logics has been one of the prominent topics in finite and computational model theory for decades. Prominent applications of model checking include database systems or computer aided verification, where logical methods are prevalent.

For classical logics such as first-order logic (FO) or monadic second-order logic (MSO) the model checking problem is known to be computationally intractable: it is PSPACE-complete for FO and MSO. These hardness results even hold in restriction to structures with only two

* Stephan Kreutzer's research has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (ERC Consolidator Grant DISTRUCT, grant agreement No 648527).

elements and no relations. As in most "model centric" contexts two element structures are not particularly interesting, it has become standard to measure the complexity of model checking problems in a more refined complexity framework, especially in the framework of parameterized complexity.

In the parameterized setting, the goal is to develop algorithms deciding $\mathrm{MC}(\mathcal{L},\mathcal{C})$ on input $\varphi \in \mathcal{L}$ and $\mathfrak{A} \in \mathcal{C}$ in time $f(|\varphi|) \cdot |\mathfrak{A}|^c$, where f is a computable function and c is a constant. Problems that can be solved in this time are called *fixed-parameter tractable* (fpt). If, furthermore, $c = 1$ they are called *fixed-parameter linear*.

If $\mathrm{MC}(\mathcal{L},\mathcal{C})$ is fpt, then a fixed formula φ can be evaluated efficiently even in very large structures, as the running time in the size of $\mathfrak{A}$ is only linear or polynomial. This is not only a theoretical observation but seems to be reflected also in practical applications. For instance, model checking for one of the widely used temporal logics, the *linear time logic* (LTL), is PSPACE-complete. But it is fixed-parameter tractable and it can be solved very efficiently in real world applications.

For logics such as MSO or FO, it is not hard to see that their model checking problem is not fixed-parameter tractable in general.[1] On the other hand, Courcelle [2] showed that if $\mathcal{C}$ is a class of structures of bounded *tree width*, then $\mathrm{MC}(\mathrm{MSO},\mathcal{C})$ is fixed-parameter linear, even if quantification over sets of edges as well as sets of vertices is allowed (this logic is called MSO_2 whereas MSO_1 only has quantification over sets of vertices in a graph).

MSO_2 is a very powerful language in which many natural graph theoretical problems can be formulated easily and naturally. It has attracted significant interest in the parameterized graph algorithms community, especially in a field known as algorithmic graph structure theory where researchers develop algorithms for NP-hard problems that become efficient for special classes of graphs, such as classes with forbidden minors. Initially, Courcelle's theorem served as a quick and easy way of proving that a problem is linear time solvable for classes of bounded tree width. Subsequently it has become a valuable tool in parameterized algorithms research for proving general meta-tractability results such as meta-kernels and the core concepts of the model-theoretic proof of Courcelle's theorem have found its way into the standard tool set of parameterized graph algorithmics, e.g. in the form of *protrusions*. Furthermore, Langer et al. [16] showed that an MSO_2-solver can be implemented and used for solving real-world problems. Hence, MSO_2 can now even be used as a high-level programming language for graph problems.

As mentioned above, for logics such as FO or MSO, $\mathrm{MC}(\mathcal{L},\mathcal{C})$ is (presumably) not fixed-parameter tractable on the class of all finite structures or graphs but it is tractable if $\mathcal{C}$ has bounded tree width. This raises the natural question for a logic $\mathcal{L}$ such as $\mathrm{FO}, \mathrm{MSO}_1, \mathrm{MSO}_2$:

> *What are the largest classes $\mathcal{C}$ of graphs or structures for which* $\mathrm{MC}(\mathcal{L},\mathcal{C})$ *is still fixed-parameter tractable.*

Or:

> *Can we identify a structural property P such that* $\mathrm{MC}(\mathcal{L},\mathcal{C})$ *is fixed-parameter tractable for a class $\mathcal{C}$ if, and only if, $\mathcal{C}$ has property P?*

Such a characterisation (of course subject to complexity theoretic assumptions) would be extremely interesting as it

[1] Here and elsewhere in this abstract we assume the standard hypotheses in complexity theory such as $\mathrm{P} \neq \mathrm{NP}$ and a similar assumption in parameterized complexity theory and our hardness results are subject to these assumptions.

(a) completely explains what makes model checking for this logic hard,
(b) identifies a set of algorithmic techniques that can be employed in the tractable cases to evaluate formulas quickly, and
(c) provided that the property P is a natural and efficiently decidable parameter, yields a quick way of checking whether model checking for the logic $\mathcal{L}$ is a particular application determined by the class $\mathcal{C}$ of structures is feasible.

This leads to a systematic research programme aiming at identifying such characterisations of tractable cases in terms of parameters P for the major logics such as $\mathrm{FO}, \mathrm{MSO}_1$ and MSO_2.

Following Courcelle's theorem, this research programme has received significant attention in the area. In [15, 14] it was proved in that fixed-parameter tractability of MSO_2 cannot be extended much beyond the case of bounded tree width. This does not establish a tight characterisation but shows that tree width seems to be the characterising parameter for MSO_2 tractability.

For MSO_1, i.e. monadic second-order logic with quantification over sets of vertices only, it was proved by Courcelle et al. [3] that $\mathrm{MC}(\mathrm{MSO}_1, \mathcal{C})$ is fixed-parameter tractable on any class $\mathcal{C}$ of graphs of bounded clique width. Here, no matching lower bound is known and to date the right graph theoretical tools to establish such a bound are still missing.

Most research, however, has gone into understanding the tractable cases for first-order model checking. For a long time much of this research has focussed on sparse classes of graphs[2]. Seese [20] showed that $\mathrm{MC}(\mathrm{FO}, \mathcal{C})$ is fpt for classes $\mathcal{C}$ of graphs of bounded maximum degree. The result was generalised by Flum and Grohe to classes with excluded minors [7] and to classes of bounded local tree width by Flum, Frick and Grohe [6]. This was followed by a series of related results, e.g. in [5, 4, 21]. See also [11] for a survey of earlier work in this area.

Finally, the case for sparse classes of graphs was settled completely in [12], where it was proved that if $\mathcal{C}$ is a class of graphs that is closed under taking subgraphs then $\mathrm{MC}(\mathrm{FO}, \mathcal{C})$ is fixed-parameter tractable if, and only if, $\mathcal{C}$ is *nowhere dense*. Nowhere dense classes of graphs have been introduced by Nešetřil and Ossonda de Mendez [18, 19, 17] as a model for *sparseness*. A huge number of results that have been obtained in recent years support their claim that nowhere denseness captures structural sparsity.

The result in [12] completely characterises the tractable cases of first-order model checking for classes of structures closed under substructures by a simple and natural parameter. Consequently, current research activities focus on model checking on dense classes of graphs. For instance, Ganian et al. [10] showed that first-order model checking is fpt on certain classes of interval graphs. And Bova et al. [1] proved that model checking for existential FO is fpt on classes of partial orders. This was later generalised by Gajarsky et al. [8] to full FO.

These examples demonstrate tractability of first-order model checking on classes of graphs that were already well established and studied in different contexts.

A different approach is to study transformations of graph classes that preserve efficient model checking. The most notably of these are first-order interpretations. A currently very active topic in this field is to study the closure of sparse classes of graphs under first-order

[2] We call a class of graphs *sparse* if the number of edges is essentially linear in the number of vertices. There are several mathematical concepts trying to define sparseness formally, including *bounded average degree*, *degeneracy* or *nowhere dense classes* of graphs. Usually, sparse classes of graphs can be closed under taking subgraphs, i.e. if $G \in \mathcal{C}$ and $H \subseteq G$ then $H \in \mathcal{C}$. The obvious exception is bounded average degree.

interpretations designed in a way that the tractability of first-order model checking is carried over from the sparse to the dense class. This route was for instance taken by Gajarsky et al. who studied the interpretation closure of classes of graphs of bounded degree and obtained a very elegant solution to this problem. See e.g. [9].

For the sparse setting, model checking for FO was studied on classes of graphs closed under taking subgraphs. This is a perfectly natural operation in the sparse setting. For dense graphs, it seems equally natural to consider classes of graphs closed under induced subgraphs. A third approach therefore is to study graph parameters which for sparse classes of graphs are equivalent to nowhere denseness but are also well defined for classes closed under induced subgraphs. One such parameter is *stability*. Stable classes of graphs hold the promise to be a very interesting class for model checking and other algorithmic purposes. See e.g. [13]. But there is still a lot of work required to establish their basic algorithmic properties.

While currently there is significant progress and activity in the study of first-order model checking on dense classes of graphs, we are still very far from a precise characterisation of the dense classes of graphs with tractable model checking.

In this talk we will briefly review the results obtained for model-checking problems in the sparse setting and then present the new ideas and results and future directions for first-order model checking on dense classes of graphs.

References

1. Simone Bova, Robert Ganian, and Stefan Szeider. Model checking existential logic on partially ordered sets. *ACM Trans. Comput. Log.*, 17(2):10:1–10:35, 2016. doi:10.1145/2814937.
2. B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, pages 194–242. Elsevier, 1990.
3. Bruno Courcelle, Janos Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.
4. A. Dawar, M. Grohe, and S. Kreutzer. Locally excluding a minor. In *Logic in Computer Science (LICS)*, pages 270–279, 2007.
5. Zdeněk Dvořák, Daniel Král, and Robin Thomas. Deciding first-order properties for sparse graphs. In *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.
6. Jörg Flum, Markus Frick, and Martin Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6):716–752, 2002. doi:10.1145/602220.602222.
7. Jörg Flum and Martin Grohe. Fixed-parameter tractability, definability, and model checking. *SIAM Journal on Computing*, 31:113–145, 2001.
8. Jakub Gajarský, Petr Hlinený, Daniel Lokshtanov, Jan Obdrzálek, Sebastian Ordyniak, M. S. Ramanujan, and Saket Saurabh. FO model checking on posets of bounded width. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 963–974, 2015. doi:10.1109/FOCS.2015.63.
9. Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, Daniel Lokshtanov, and M. S. Ramanujan. A new perspective on FO model checking of dense graph classes. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'16, New York, NY, USA, July 5-8, 2016*, pages 176–184, 2016. doi:10.1145/2933575.2935314.
10. Robert Ganian, Petr Hlinený, Daniel Král, Jan Obdrzálek, Jarett Schwartz, and Jakub Teska. FO model checking of interval graphs. *Logical Methods in Computer Science*, 11(4), 2015. doi:10.2168/LMCS-11(4:11)2015.

11 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta-theorems. *Contemporary Mathematics*, 588, American Mathematical Society 2011.

12 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. In *46th Annual Symposium on the Theory of Computing (STOC)*, 2014.

13 Stephan Kreutzer, Roman Rabinovich, and Sebastian Siebertz. Polynomial kernels and wideness properties of nowhere dense graph classes. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1533–1545, 2017. doi:10.1137/1.9781611974782.100.

14 Stephan Kreutzer and Siamak Tazari. Lower bounds for the complexity of monadic second-order logic. In *Logic in Computer Science (LICS)*, 2010.

15 Stephan Kreutzer and Siamak Tazari. On brambles, grid-like minors, and parameterized intractability of monadic second-order logic. In *Symposium on Discrete Algorithms (SODA)*, 2010.

16 Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Evaluation of an mso-solver. In *Algorithm Engineering & Experiments (ALENEX)*, pages 55–63, 2012.

17 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity*. Springer, 2012.

18 Jaroslav Nešetřil and Patrice Ossona de Mendez. First order properties of nowhere dense structures. *Journal of Symbolic Logic*, 75(3):868–887, 2010.

19 Jaroslav Nešetřil and Patrice Ossona de Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

20 Detlev Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 5:505–526, 1996.

21 Luc Segoufin and Alexandre Vigny. Constant Delay Enumeration for FO Queries over Databases with Local Bounded Expansion. In Michael Benedikt and Giorgio Orsi, editors, *20th International Conference on Database Theory (ICDT 2017)*, volume 68 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICDT.2017.20.

Arithmetic Circuits: An Overview

Meena Mahajan

The Institute of Mathematical Sciences, HBNI, Chennai, India
meena@imsc.res.in

Abstract

This talk reviews recent developments in algebraic complexity theory. It outlines some major results concerning structure, completeness, closure, and lower bounds. It describes some techniques that have been central to obtaining these results, including extreme depth reduction, partial derivatives, and padding.

Some recent surveys on arithmetic circuits appear in [4] and [1]. A continuously updated online survey on lower bounds appears at [3].

1998 ACM Subject Classification F.1.1 Models of Computation/Circuits, F.1.3 Complexity Measures and Classes

Keywords and phrases algebraic complexity, circuits, formulas, branching programs, determinant, permanent

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.5

Category Invited Talk

References

1 M. Mahajan. Algebraic complexity classes. In *Perspectives in Computational Complexity: The Somenath Biswas Anniversary Volume*, pages 51–75. Birkhäuser, 2014.

2 Meena Mahajan and Nitin Saurabh. Some complete and intermediate polynomials in algebraic complexity theory. *Thory of Computing Systems*, page to appear, 2017. (CSR special issue). doi:10.1007/s00224-016-9740-y.

3 Ramprasad Saptharishi. A survey of known lower bounds in arithmetic circuits. A continuously updated git survey. URL: https://github.com/dasarpmar/lowerbounds-survey.

4 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

Determinacy of Infinite Games: Perspectives of the Algorithmic Approach

Wolfgang Thomas

RWTH Aachen University, Aachen, Germany
thomas@cs.rwth-aachen.de

Abstract

Determinacy of infinite two-player games is a topic of descriptive set theory that has triggered intensive research in theoretical computer science since 1957 when A. Church formulated his "synthesis problem" (regarding the construction of circuits with infinite behavior from logical specifications). In the first part of the lecture we review the fascinating development of the algorithmic theory of infinite games that was started by Church's problem, that enriched automata theory and related fields, and that led to interesting applications in verification and program synthesis. In the second part we turn to the question how to lift this theory from the case of the Cantor space (where a play is a sequence of bits) to the case of the Baire space (where a play is a sequence of natural numbers). While this step does not involve difficulties in classical descriptive set theory, the algorithmic approach raises non-trivial questions since it requires to consider automata that work over infinite alphabets. We present recent results (joint work with B. Brütsch) that provide a solution of Church's synthesis problem in this context, and we point to numerous questions that are still open.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.1.1 Models of Computation

Keywords and phrases Infinite games, descriptive set theory, automata theory, transducers, automatic synthesis

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.6

Category Invited Talk

Symbolic Automata Theory with Applications

Margus Veanes

Microsoft Research, Redmond, WA, USA
margus@microsoft.com

Abstract

Symbolic automata extend classic finite state automata by allowing transitions to carry predicates over rich alphabet theories. The key algorithmic difference to classic automata is the ability to efficiently operate over very large or infinite alphabets. In this talk we give an overview of what is currently known about symbolic automata, what their main applications are, and what challenges arise when reasoning about them. We also discuss some of the open problems and research directions in symbolic automata theory.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.3 Formal Languages

Keywords and phrases automata, transducers, infinite alphabets

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.7

Category Invited Talk

1 Overview

This talk provides an overview of the recent results in the theory and practice of *symbolic automata*, which are models for automata based reasoning about sequences and trees over complex element domains. Classic finite state automata have been used in a wide variety of applications, including lexical analysis [1], software verification [3], text processing [2], and computational linguistics [13]. One major limitation of classic automata is that their alphabets need to be finite and small for the algorithms to scale. To overcome this limitation, there have been proposals to extend classic automata to use *predicates* instead of concrete characters on state transitions [16, 20]. This talk focuses on work done following the definition of symbolic finite automata presented in [17], where predicates are drawn from an effective Boolean algebra. Other, orthogonal, approaches to accommodate infinite alphabets are based on automata with registers [12]. A meaning of symbolic automata that is different from the one used here is sometimes used to refer to classic finite state automata with BDD based representation of state spaces [14].

Despite the support for infinite alphabets, symbolic automata retain many of the good properties of their finite-alphabet counterparts, such as closure under Boolean operations. The theory and algorithms of *symbolic finite automata* (s-FAs) and *symbolic finite transducers* (s-FTs) has recently received considerable attention [18, 5]. Many applications have emerged that make use of s-FAs and s-FTs: verification of string sanitizers [10], analysis of tree-manipulating programs [9], synthesis of string encoders [11], regex support in parameterized unit testing [17], similarity analysis of binaries [4], parallelization of string manipulating code [19], and fusion of streaming computations [15].

There are also some cases when classic properties over finite alphabets, either do not generalize to the symbolic setting [6, 11], or when algorithms have turned out to be difficult to generalize to the symbolic setting [7] due to lack of proper data structure support. The intent of this talk is to explain the difference between the symbolic and the finite-alphabet case, to

give an overview about what is known in symbolic automata theory, which applications have been the driving force behind this theory, and to discuss open problems. A recent tutorial on symbolic automata and transducers is given in [8] that also presents some new properties not formally investigated in earlier papers.

References

1 Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison-Wesley, 2006.

2 Rajeev Alur, Loris D'Antoni, and Mukund Raghothaman. Drex: A declarative language for efficiently evaluating regular string transformations. *ACM SIGPLAN Notices – POPL'15*, 50(1):125–137, 2015.

3 Ahmed Bouajjani, Peter Habermehl, and Tomáš Vojnar. Abstract regular model checking. In *CAV'04*, pages 372–386. Springer, 2004.

4 Mila Dalla Preda, Roberto Giacobazzi, Arun Lakhotia, and Isabella Mastroeni. Abstract symbolic automata: Mixed syntactic/semantic similarity analysis of executables. *ACM SIGPLAN Notices – POPL'15*, 50(1):329–341, 2015.

5 Loris D'Antoni and Margus Veanes. Minimization of symbolic automata. *ACM SIGPLAN Notices – POPL'14*, 49(1):541–553, 2014.

6 Loris D'antoni and Margus Veanes. Extended symbolic finite automata and transducers. *Formal Methods System Design*, 47(1):93–119, August 2015.

7 Loris D'Antoni and Margus Veanes. Forward bisimulations for nondeterministic symbolic finite automata. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 23rd International Conference, TACAS 2017*, volume 10205 of *LNCS*, pages 518–534. Springer, 2017.

8 Loris D'Antoni and Margus Veanes. The power of symbolic automata and transducers. In *Computer Aided Verification, 29th International Conference (CAV'17)*. Springer, 2017.

9 Loris D'Antoni, Margus Veanes, Benjamin Livshits, and David Molnar. Fast: A transducer-based language for tree manipulation. *ACM TOPLAS*, 38(1):1–32, 2015.

10 Pieter Hooimeijer, Benjamin Livshits, David Molnar, Prateek Saxena, and Margus Veanes. Fast and precise sanitizer analysis with BEK. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, 2011.

11 Qinheping Hu and Loris D'Antoni. Automatic program inversion using symbolic transducers. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2017, pages 376–389, 2017.

12 Amaldev Manuel and Ramaswamy Ramanujam. Automata over infinite alphabets. In Deepak D'Souza and Priti Shankar, editors, *Modern Applications of Automata Theory*, pages 529–554. World Scientific, 2012.

13 Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

14 Kristin Y. Rozier and Moshe Y. Vardi. A multi-encoding approach for LTL symbolic satisfiability checking. In *FM'11*, pages 417–431, 2011.

15 Olli Saarikivi, Margus Veanes, Todd Mytkowicz, and Madan Musuvathi. Fusing effectful comprehensions. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2017, pages 17–32. ACM, 2017.

16 Gertjan van Noord and Dale Gerdemann. Finite state transducers with predicates and identities. *Grammars*, 4(3):263–286, 2001.

17 Margus Veanes, Peli de Halleux, and Nikolai Tillmann. Rex: Symbolic regular expression explorer. In *ICST'10*, pages 498–507. IEEE, 2010.

18 Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjørner. Symbolic finite state transducers: Algorithms and applications. *ACM SIGPLAN Notices – POPL'12*, 47(1):137–150, 2012.

19 Margus Veanes, Todd Mytkowicz, David Molnar, and Benjamin Livshits. Data-parallel string-manipulating programs. *ACM SIGPLAN Notices – POPL'15*, 50(1):139–152, 2015.

20 Bruce W. Watson. *Extended Finite State Models of Language*, chapter Implementing and using finite automata toolkits, pages 19–36. Cambridge U. Press, 1999.

Categorical Structures for Type Theory in Univalent Foundations*

Benedikt Ahrens[1], Peter LeFanu Lumsdaine[2], and Vladimir Voevodsky[3]

1 **Inria, Nantes, France**
`benedikt.ahrens@inria.fr`
2 **Department of Mathematics, Stockholm University, Stockholm, Sweden**
`p.l.lumsdaine@math.su.se`
3 **Institute for Advanced Study, Princeton, NJ, USA**
`vladimir@ias.edu`

Abstract

In this paper, we analyze and compare three of the many algebraic structures that have been used for modeling dependent type theories: *categories with families*, *split type-categories*, and *representable maps of presheaves*. We study these in the setting of univalent foundations, where the relationships between them can be stated more transparently. Specifically, we construct maps between the different structures and show that these maps are equivalences under suitable assumptions.

We then analyze how these structures transfer along (weak and strong) equivalences of categories, and, in particular, show how they descend from a category (not assumed univalent/saturated) to its Rezk completion. To this end, we introduce *relative universes*, generalizing the preceding notions, and study the transfer of such relative universes along suitable structure.

We work throughout in (intensional) dependent type theory; some results, but not all, assume the univalence axiom. All the material of this paper has been formalized in Coq, over the UniMath library.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages, F.4.1 Mathematical Logic

Keywords and phrases Categorical Semantics, Type Theory, Univalence Axiom

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.8

1 Introduction

Various kinds of categorical structures have been introduced in which to interpret type theories: for instance, categories with families, C-systems, categories with attributes, and

* This material is based upon work supported by the National Science Foundation under agreement Nos. DMS-1128155 and CMU 1150129-338510. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This material is based on research sponsored by The United States Air Force Research Laboratory under agreement number FA9550-15-1-0053. The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Air Force Research Laboratory, the U.S. Government or Carnegie Mellon University. This work has partly been funded by the CoqHoTT ERC Grant 637339, and by the Swedish Research Council (VR) Grant 2015-03835 *Constructive and category-theoretic foundations of mathematics*.

so on. The aim of these structures is to encompass and abstract away the structural rules of type theories – weakening, substitution, and so on – that are independent of the specific logical type and term formers of the type theory under consideration.

For a given kind of categorical structures and a given type theory one expects furthermore to be able to build, from the syntax, an initial object of a category or 2-category whose objects are categorical structures of this kind equipped with suitable extra operations 'modelling' the type and term formers of the type theory under consideration. This universal property then provides a means to construct interpretations of the syntax, by assembling the objects of the desired interpretation into another instance of such a structure.

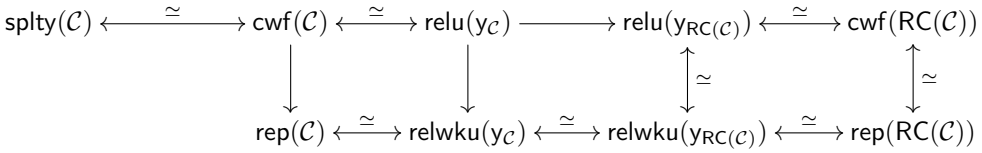
It is natural to ask if, and in what sense, these various categorical structures are equivalent, or otherwise related. The equivalences, differences, and comparisons between them are often said to be well-known, but few precise statements exist in the literature.

The goals of the present work are twofold. Firstly, to give some such comparisons precisely and carefully. And secondly, to illustrate how in univalent foundations, such comparisons can be approached in a different and arguably more straightforward fashion than in classical settings.

More specifically, the paper falls into two main parts:

- comparison between categories with families (CwF's) and split type-categories;
- interaction between CwF structures and Rezk completion of categories, and comparison between CwF structures and representable maps of presheaves, all via comparison with *relative universes* (introduced in the present work).

Our constructions and equivalences may be summed up in the following diagram:



All proofs and constructions of the article have been formalized in Coq, over the `UniMath` library. We take this as licence to err on the side of indulgence in focusing on the key ideas of constructions and suppressing less-enlightening technical details, for which we refer readers to the formalization. An overview the formalization is given in Section 6. Throughout, we annotate results with their corresponding identifier in the formalization, in `teletype_font`.

2 Background

2.1 Categorical structures for type theory

In this short section we briefly review some of the various categorical structures for dependent type theory introduced in the literature. We do not aim to give a comprehensive survey of the field, but just to recall what pertains to the present paper.

The first introduced were Cartmell's *contextual categories* [4, 5], since studied by Voevodsky under the name of *C-systems* [15, 13].

Pitts [10, Def. 6.3.3] introduced *split type-categories*, originally just as *type-categories*; Van den Berg and Garner [12, Def. 2.2.1] (whom we follow here) later reorganized Pitts' definition to isolate the splitness conditions, and hence allow what they call *(non-split) type-categories*. These have also been studied as *categories with attributes* by Hofmann [8] and others.[1]

[1] Cartmell originally used the name *categories with attributes* for a slightly different notion [4, §3.2].

Categories with families were defined by Dybjer [6] to make explicit the data of *terms*, not taken as primary in the previous approaches. A functional variant of Dybjer's definition, which we follow in the present paper, was suggested by Fiore [7, Appendix], and studied under the name *natural models* by Awodey [3] (see also footnote to Def. 32 below).

The notions of *universes* and *universe categories*, which we generalize in the present work to *universes relative to a functor*, were introduced by Voevodsky in [13, Def. 2.1].

2.2 The agnostic, univalent, and classical settings

The background setting of the present work is intensional type theory, assuming throughout: Σ-types, with the strong η rule; identity types; Π-types, also with η, and functional extensionality; $\mathsf{0}$, $\mathsf{1}$, $\mathsf{2}$, and N; propositional truncation (axiomatised as in [11, §3.7]); and two universes closed under all these constructions.

All the above is *agnostic* about equality on types – it is not assumed either to be univalent, or to be always a proposition – and hence is expected to be compatible with the interpretation of types as classical sets, as well as homotopical interpretations such as the simplicial set model. In particular, our main definitions of categorical structures use only this background theory, and under the classical interpretation they become the established definitions from the literature. Similarly, most of the comparison maps we construct rely only on this, so can be understood in the classical setting.

Other results, however – essentially, all non-trivial equivalences of types we prove – assume additionally the univalence axiom; these therefore hold only in the univalent setting, and are not compatible with the classical interpretation.

We mostly follow the type-theoretic vocabulary standardized in [11]. A brief, but sufficient, overview is given in [1], among other places. By Set we denote the category of h-sets of a fixed, but unspecified universe.

We depart from it (and other type-theoretic traditions) in using *existence* for what is called *mere existence* in [11] and *weak existence* by Howard [9], since this is what corresponds to the standard mathematical usage of *existence*.

2.3 Comparing structures in the univalent setting

Suppose one wishes to show that two kinds of mathematical object are equivalent – say, one-object groupoids and groups. What precise statement should one aim for?

In the classical setting, the most obvious candidate – a bijection of the sets (or classes) of these objects – is not at all satisfactory. On the one hand, the natural back-and-forth constructions may well fail to form a bijection. On the other hand, the axiom of choice may imply that bijections exist even when the objects involved are quite unrelated.

To give a more meaningful statement, one may define suitable morphisms, corral the objects into two categories, and construct an equivalence of categories between these. Sometimes, one needs to go further, and construct an equivalence of higher categories, or spaces.

In the univalent setting, however, life is simpler. The most straightforward candidate, an equivalence of *types* between the types of the two kinds of objects, is already quite satisfactory and meaningful, corresponding roughly to an equivalence between the *groupoids* of such objects in the classical setting (or higher groupoids, etc.).

This is not to dismiss the value given by constructing (say) an equivalence of categories However, defining the morphisms and so on is no longer *required* in order to give a meaningful equivalence between the two kinds of objects.

Another advantage of a comparison in terms of equivalence of types is its *uniformity*. Indeed, the one notion of equivalence of types can serve to compare objects that naturally form the elements of sets, or the objects of categories, or bicategories, etc.

In the present paper, therefore, we take advantage of this: two of our main results are such equivalences, between the types of categories with families and of split type-categories, and between the types of representable maps of presheaves on a category $\mathcal{C}$, and of CwF structures on its Rezk completion.

(We have focused here on equivalences, but the principle applies equally for other comparisons: for instance, an injection of types carries more information than an injection of sets/classes, corresponding roughly to a full and faithful map of (possibly higher) groupoids.)

2.4 Categories in the univalent setting

The fundamentals of category theory were transferred to the univalent setting in [1]. Two primary notions of category are defined, there called *precategories* and *categories*. We change terminology, calling their precategories *categories* (since it is this that becomes the traditional definition under the set interpretation), and their categories *univalent categories*.

Specifically, a *category* $\mathcal{C}$ (in our terminology) consists of:

- a type $\mathcal{C}_0$, its *objects*;
- for each $a, b : \mathcal{C}_0$, a set $\mathcal{C}(a, b)$, the *morphisms* or *maps* from a to b;
- together with identity and composition operations satisfying the usual axioms.

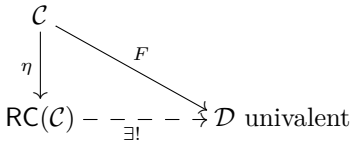
We emphasize that the hom-sets $\mathcal{C}(a, b)$ are required to be sets, but $\mathcal{C}_0$ is allowed to be an arbitrary type.

In any category $\mathcal{C}$, there is a canonical map from equalities of objects to isomorphisms, $\mathsf{idtoiso}_{a,b} : (a =_{\mathcal{C}_0} b) \to \mathsf{Iso}_{\mathcal{C}}(a, b)$. We say that $\mathcal{C}$ is *univalent* if for all $a, b : \mathcal{C}_0$, this map $\mathsf{idtoiso}_{a,b}$ is an equivalence: informally, if 'equality of objects is isomorphism'.

A central example is the category Set of sets (in some universe). Univalence of this category follows directly from the univalence axiom for the corresponding universe. It follows in turn that $\mathsf{PreShv}(\mathcal{C})$, the category of presheaves on a category $\mathcal{C}$, is also always univalent. We write $\mathsf{y}_{\mathcal{C}} : \mathcal{C} \to \mathsf{PreShv}(\mathcal{C})$ for the Yoneda embedding.

In properties of functors, we distinguish carefully between existence properties and chosen data. We say a functor $F : \mathcal{C} \to \mathcal{D}$ is *essentially surjective* if for each $d : \mathcal{D}_0$, there exist some $c : \mathcal{C}_0$ and isomorphism $i : Fc \cong d$, and is *split essentially surjective* if it is equipped with an operation giving, for each $d : \mathcal{D}_0$, some such c and i. A *weak equivalence* is a functor that is full, faithful, and essentially surjective.

An important construction from [1] is the *Rezk completion* $\mathsf{RC}(\mathcal{C})$ of a category $\mathcal{C}$, the 'free' univalent category on $\mathcal{C}$. Precisely, $\mathsf{RC}(\mathcal{C})$ is univalent, and there is a weak equivalence $\eta_{\mathcal{C}} : \mathcal{C} \to \mathsf{RC}(\mathcal{C})$, enjoying the expected universal property: any functor from $\mathcal{C}$ to a univalent category factorizes uniquely through $\eta_{\mathcal{C}}$.



Note that the main definitions make sense in the agnostic background setting, and under the classical interpretation become the standard definitions; but the Rezk completion construction and the univalence of Set and $\mathsf{PreShv}(\mathcal{C})$ rely additionally on the univalence axiom.

Indeed, [1] additionally assumes *resizing axioms*, in order to show that $\mathsf{RC}(\mathcal{C})$ is no larger than $\mathcal{C}$ That is, a priori, the type $\mathsf{RC}(\mathcal{C})_0$ lives in a higher universe than the type $\mathcal{C}_0$. However,

for the present paper, this size issue is not a problem; so we do not need to assume resizing rules.

As usual, we will write $f : a \to b$ for $f : \mathcal{C}(a, b)$ in arbitrary categories, and will write $c : \mathcal{C}$ rather than $c : \mathcal{C}_0$. We write composition in the 'diagrammatic' order; that is, the composite of $f : a \to b$ and $g : b \to c$ is denoted $f \cdot g : a \to c$.

3 Equivalence between categories with families and split type-categories

3.1 CwF's and type-categories

For the most part, we take care to follow established definitions closely. We depart from most literature however in one way: we do not take CwF's or type-categories (or other similar structures) to include a terminal object. This does not interact in any way with the rest of the structure, so does not affect the equivalences we construct below. We do this since both versions (with and without terminal object) seem useful for the development of the theory; and it is easier to equip objects with extra structure later than to remove it.

▶ **Definition 1** ([7, Appendix]; `cwf_structure`, `cwf`)**.** A *category with families (à la Fiore)* consists of:

0. a category $\mathcal{C}$, together with
1. presheaves $\mathsf{Ty}, \mathsf{Tm} : \mathcal{C}^{\mathrm{op}} \to \mathsf{Set}$;
2. a natural transformation $\mathsf{p} : \mathsf{Tm} \to \mathsf{Ty}$; and
3. for each object $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$, a representation of the fiber of p over A, i.e.
  a. an object $\Gamma.A : \mathcal{C}$ and map $\pi_A : \Gamma.A \to \Gamma$,
  b. an element $\mathsf{te}_A : \mathsf{Tm}(\Gamma.A)$, such that $\mathsf{p}_{\Gamma.A}(\mathsf{te}_A) = \pi_A^* A : \mathsf{Ty}(\Gamma.A)$,
  c. and such that the induced commutative square

$$\begin{array}{ccc} \mathsf{y}(\Gamma.A) & \xrightarrow{\widehat{\mathsf{te}_A}} & \mathsf{Tm} \\ {\scriptstyle \mathsf{y}(\pi_A)}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{p}} \\ \mathsf{y}(\Gamma) & \xrightarrow[\widehat{A}]{} & \mathsf{Ty} \end{array}$$

is a pullback. (Here e.g. $\widehat{A}$ denotes the Yoneda transpose of an element of a presheaf.) By a *CwF structure* on a category $\mathcal{C}$, we mean the data of items 1–3 above.

▶ Remark. This is a reformulation, due to Fiore, of Dybjer's original definition of CwF's [6, Def. 1], replacing the single functor $\mathcal{C} \to \mathsf{Fam}$ by the map of presheaves $\mathsf{p} : \mathsf{Tm} \to \mathsf{Ty}$.

▶ **Definition 2.** Let $\mathcal{C}$ be a CwF, $\Gamma : \mathcal{C}$ an object, and $e : A = B$ an equality of elements of $\mathsf{Ty}(\Gamma)$. We write $\Delta_e : \Gamma.A \cong \Gamma.B$ for the induced isomorphism $\mathsf{idtoiso}(\mathsf{ap}_{x \mapsto \Gamma.x}(e))$.

Since $\mathsf{Ty}(\Gamma)$ is a set, we will sometimes suppress e and write just $\Delta_{A,B}$. We will also use this notation in other situations with a family Ty and operation $\Gamma, A \mapsto \Gamma.A$ as in a CwF.

▶ **Definition 3** ([10]; `typecat_structure`, `is_split_typecat`, `split_typecat_structure`)**.** A *type-category* consists of:

0. a category $\mathcal{C}$, together with
1. for each object $\Gamma : \mathcal{C}$, a type $\mathsf{Ty}(\Gamma)$,
2. for each $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$, an object $\Gamma.A : \mathcal{C}$

3. for each such Γ, A, a morphism $\pi_A : \Gamma.A \to \Gamma$,
4. for each map $f : \Gamma' \to \Gamma$, a 'reindexing' function $\mathsf{Ty}(\Gamma) \to \mathsf{Ty}(\Gamma')$, denoted $A \mapsto f^*A$,
5. for each Γ, $A : \mathsf{Ty}(\Gamma)$, and $f : \Gamma' \to \Gamma$, a morphism $\mathsf{q}(f, A) : \Gamma'.f^*A \to \Gamma.A$,
6. such that for each such Γ, A, Γ', f, the following square commutes and is a pullback:

$$\begin{array}{ccc} \Gamma'.f^*A & \xrightarrow{\mathsf{q}(f,A)} & \Gamma.A \\ {\scriptstyle \pi_{f^*A}}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi_A} \\ \Gamma' & \xrightarrow[f]{} & \Gamma \end{array}$$

A type-category is *split* if:

7. for each Γ, the type $\mathsf{Ty}(\Gamma)$ is a set;
8. for each Γ and $A : \mathsf{Ty}(\Gamma)$, we have equalities
   a. $e : 1_\Gamma^* A = A$, and
   b. $\mathsf{q}(1_\Gamma, A) = \Delta_e \ : \ \Gamma.1_\Gamma^* A \to \Gamma.A$; and
9. for $f' : \Gamma'' \to \Gamma')$, $f : \Gamma' \to \Gamma$, and $A : \mathsf{Ty}(\Gamma)$, we have equalities
   a. $e' : (f' \cdot f)^* A = f'^* f^* A$, and
   b. $\mathsf{q}(f' \cdot f, A) = \Delta_{e'} \cdot \mathsf{q}(f', f^*A) \cdot \mathsf{q}(f, A) \ : \ \Gamma''.(f' \cdot f)^* A \to \Gamma.A$.

In the present work, we will only consider split type-categories. Non-split type-categories are however also of great importance, especially in the agnostic/univalent settings, since classical methods for constructing split ones may no longer work.

3.2 Equivalence between CwF's and split type-categories

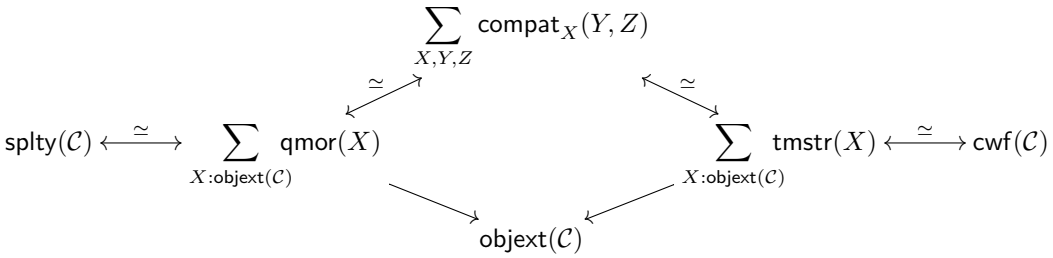
The main goal of this section is to construct an equivalence of types between the type of CwF's and the type of split type-categories. In outline, we proceed as follows:

Firstly, we specialize to giving an equivalence between CwF structures and split type-category structures over a fixed base category $\mathcal{C}$.

Secondly, we further abstract out the shared part of these, decomposing them into

- *object extension structures*, the shared structure common to CwF's and split type-categories, and
- structures comprising the remaining data of CwF structures and split type-category structures, which we call *term-structures* and q-*morphism structures* respectively.

Finally, we give an equivalence between term-structures and q-morphism structures over a given category and object extension structure. We do this by defining a *compatibility* relation between them, and showing that for each term-structure there exists a unique compatible q-morphism structure, and vice versa.



▶ **Definition 4** (`obj_ext_structure`). A *(split) object extension structure* on a category $\mathcal{C}$ consists of:

1. a functor $\mathsf{Ty} : \mathcal{C}^{\mathrm{op}} \to \mathsf{Set}$;
2. for each $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$,
   a. an object $\Gamma.A$, and
   b. a *projection* morphism $\pi_A : \mathcal{C}(\Gamma.A, \Gamma)$.

In general, one might want to loosen the setness and functoriality conditions on Ty, and hence distinguish the present definition as the split version of a more general notion. In this paper, however, we do not consider the non-split case, so take *object extension structures* to mean the split ones throughout.

▶ **Definition 5** (`term_fun_structure`)**.** Let $\mathcal{C}$ be a category equipped with an object extension structure X. A *(functional) term-structure* over X consists of:

1. a presheaf $\mathsf{Tm} : \mathcal{C}^{\mathrm{op}} \to \mathsf{Set}$, and natural transformation $\mathsf{p} : \mathsf{Tm} \to \mathsf{Ty}$;
2. for each object $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$, an element $\mathsf{te}_A : \mathsf{Tm}(\Gamma.A)$, such that $(\Gamma.A, \pi_A, \mathsf{te}_A)$ form a representation of the fiber of p over A as in item 3 of Def. 1.

One might say *functional* to distinguish these from *familial* term-structures, which would correspond analogously to CwF's in the sense of Dybjer, with $\mathsf{Tm}(\Gamma)$ a family indexed by $\mathsf{Ty}(\Gamma)$. For the present paper, however, we work only with the functional ones, so call these simply *term-structures*.

▶ **Problem 6.** *Given a category $\mathcal{C}$, to construct an equivalence between CwF structures on $\mathcal{C}$ and pairs (X, Y) of an object extension structure X on $\mathcal{C}$ and a term-structure Y over X.*

▶ **Construction 7** (for Problem 6; `weq_cwf_cwf'_structure`)**.** Mathematically, this is essentially trivial, as is visibly evident from the definitions: just a matter of reordering and reassociating Σ-types, and distributing Π-types over Σ-types. It is perhaps worth noting however that this distributivity requires functional extensionality. ◀

▶ **Definition 8** (`qq_morphism_structure`)**.** Let $\mathcal{C}$ be a category equipped with an object extension structure X. A *(split)* q*-morphism structure* over X consists of:

1. for each $f : \Gamma' \to \Gamma$ and $A : \mathsf{Ty}(\Gamma)$, a map $\mathsf{q}(f, A) : \Gamma.A \to \Gamma$, such that following square commutes and is a pullback

$$\begin{array}{ccc} \Gamma'.f^*A & \xrightarrow{\mathsf{q}(f,A)} & \Gamma.A \\ {\scriptstyle \pi_{f^*A}}\downarrow & & \downarrow{\scriptstyle \pi_A} \\ \Gamma' & \xrightarrow[f]{} & \Gamma \end{array}$$

   and such that
2. for each Γ and $A : \mathsf{Ty}(\Gamma)$, $\mathsf{q}(1_\Gamma, A) = \Delta_{1_\Gamma^* A, A} \ : \ \Gamma.1_\Gamma^* A \to \Gamma.A$; and
3. for each $f' : \Gamma'' \to \Gamma'$, $f : \Gamma' \to \Gamma$, and $A : \mathsf{Ty}(\Gamma)$,

$$\mathsf{q}(f' \cdot f, A) = \Delta_{(f'\cdot f)^* A, f'^* f^* A} \cdot \mathsf{q}(f', f^* A) \cdot \mathsf{q}(f, A) \ : \ \Gamma''.(f' \cdot f)^* A \to \Gamma.A.$$

   Here the suppressed equalities on the Δ terms come from the functoriality axioms of Ty.

▶ **Problem 9.** *Given a category $\mathcal{C}$, to construct an equivalence between split type-category structures on $\mathcal{C}$ and pairs (X, Z) of an object extension structure X on $\mathcal{C}$ and a q-morphism structure Z over X.*

▶ **Construction 10** (for Problem 9; `weq_standalone_to_regrouped`)**.** Much like Construction 7, simply a matter of wrangling Π- and Σ-types. ◀

For most of the remainder of this section, we fix a category $\mathcal{C}$ and object extension structure X on $\mathcal{C}$. We can now explicitly define constructions going back and forth between term-structures and q-morphism structures over X, preparatory to showing that they form an equivalence. Before we do so, we define the *compatibility* relation between them.

▶ **Definition 11.** Let Y be a term-structure and Z a q-morphism structure over X. Say that Y and Z are *compatible* if for all $f : \Gamma \to \Gamma'$ and $A : \mathsf{Ty}(\Gamma)$, $\mathsf{te}_{f^*A} = \mathsf{q}(f, A)^*\mathsf{te}_A$.

▶ **Problem 12.** *Given a term-structure Y over X, to construct a q-morphism structure over X compatible with Y.*

▶ **Construction 13** (for Problem 12; `compatible_qq_from_term`)**.** Given Y and $f : \Gamma \to \Gamma'$ and $A : \mathsf{Ty}(\Gamma)$, the term-structure axioms give a pullback square

$$\begin{array}{ccc} \mathcal{C}(\Gamma'.f^*A, \Gamma.A) & \xrightarrow{g \mapsto g^*\mathsf{te}_A} & \mathsf{Tm}(\Gamma'.f^*A) \\ {\scriptstyle -\cdot\pi_A}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{p}_{\Gamma'.f^*A}} \\ \mathcal{C}(\Gamma'.f^*A, \Gamma) & \xrightarrow{g \mapsto g^*A} & \mathsf{Ty}(\Gamma'.f^*A) \end{array}$$

So we may take $\mathsf{q}(f, A)$ to be the unique map $\Gamma'.f^*A \to \Gamma.A$ such that $\mathsf{q}(f, A) \cdot \pi_A = \pi_{f^*A} \cdot f$ and $\mathsf{q}(f, A)^*\mathsf{te}_A = \mathsf{te}_{f^*A}$. Verification that this forms a compatible q-morphism structure is essentially routine calculation. ◀

▶ **Problem 14.** *Given a q-morphism structure Y over X, to construct a term-structure over X compatible with Y.*

▶ **Construction 15** (for Problem 14; `compatible_term_from_qq`)**.** This construction is rather more involved; we only sketch it here, and refer to the formalization for full details.

Briefly, $\mathsf{Tm}(\Gamma)$ is the set of pairs (A, s), where $A : \mathsf{Ty}(\Gamma)$, and $s : \Gamma \to \Gamma.A$ is a section of π_A. Its functorial action f^* involves pulling back sections along the pullback squares given by the q-morphism structure. Finally, the universal element $\mathsf{te}_A : \mathsf{Tm}(\Gamma)$ is the pair $(\pi_A^*A, \delta_{\pi_A})$, where $\delta_{\pi_A} : \Gamma.A \to \Gamma.A.\pi_A^*A$ is the diagonal map of the pullback square for A and π_A given by the q-morphism structure. ◀

▶ **Problem 16.** *(Assuming Univalence.) To give an equivalence between term-structures and q-morphism structures over X, whose underlying functions are as given in the two preceding constructions.*

From this equivalence, to derive an equivalence between the type of pairs (X, Y) of an object extension structure and a term-structure, and of pairs (X, Z) an object extension structure and a q-morphism structure.

▶ **Construction 17** (for Problem 16; `weq_cwf'_sty'`)**.** As intimated above, we proceed by showing that for each term-structure, the compatible q-morphism structure constructed above is in fact the *unique* compatible such structure, and vice versa.

This equivalence immediately induces an equivalence of pair types, which is the identity on the first component carrying the object extension structure.

Note in particular that for this result – unlike in the constructions above – we rely essentially on the univalence axiom. ◀

▶ **Problem 18.** *To construct an equivalence between cwf structures and split type-category structures on a category $\mathcal{C}$.*

▶ **Construction 19** (for Problem 18; `weq_sty_cwf`)**.** By composing the equivalences of Constructions 7, 10, and 17:

$$\mathsf{splty}(\mathcal{C}) \simeq \sum_{X:\mathsf{object}(\mathcal{C})} \mathsf{qmor}(X) \simeq \sum_{X:\mathsf{object}(\mathcal{C})} \mathsf{tmstr}(X) \simeq \mathsf{cwf}(\mathcal{C}) \ . \qquad \blacktriangleleft$$

The back-and-forth constructions above (though not compatibility) are also sketched in e.g. [8, Sections 3.1, 3.2] (note that Hofmann's *categories with attributes* are what we call *split type-categories*). Hofmann also mentions that – in our terminology – going from q-morphism structures to term-structures and back yields the original q-morphism structure. However, working in set-theoretic foundations, the same is not true for the other direction – the original term presheaf is not recovered up to equality. This is exactly where the univalence axiom comes to our rescue: it allows us to conclude that, by showing that the obtained term presheaf is *isomorphic* to the original one, the two are identical. Hence we obtain an equivalence of types. Note however that our proof that the maps are indeed an equivalence is slightly different, to avoid the difficult direct construction of an identity between two term-structures.

In the absence of the univalence axiom, the constructed maps back and forth can still be used to compare term-structures and q-morphism structures: by defining suitable notions of morphisms of those structures, these maps underlie an equivalence of categories. We have also formalized this equivalence of categories, and will report on it in a forthcoming article.

4 Relative universes and a transfer construction

In this section we introduce the notion of *universes relative to a functor* $J : \mathcal{C} \to \mathcal{D}$. This notion generalizes the universes studied in [13], and is inspired by the generalization of monads to relative monads [2].

4.1 Relative universes and weak universes

▶ **Definition 20** (`fpullback`)**.** Let $J : \mathcal{C} \to \mathcal{D}$ be a functor, and $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ a morphism of $\mathcal{D}$.

Given an object X of $\mathcal{C}$ and morphism $f : J(X) \to \mathsf{U}$ in $\mathcal{D}$, a *J-pullback of p along f* consists of an object X' of $\mathcal{C}$ and morphisms $p' : X' \to X$ and $Q : J(X') \to \tilde{\mathsf{U}}$, such that the following square commutes and is a pullback:

$$\begin{array}{ccc} J(X') & \xrightarrow{Q} & \tilde{\mathsf{U}} \\ {\scriptstyle J(p')}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{p}} \\ J(X) & \xrightarrow[f]{} & \mathsf{U} \end{array}$$

▶ **Definition 21** (`relative_universe`, `weak_relative_universe`)**.** Let $J : \mathcal{C} \to \mathcal{D}$ be a functor, as above.

A *J-universe structure* on a map $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ of $\mathcal{D}$ is a function giving, for each object X in $\mathcal{C}$ and map $f : J(X) \to \mathsf{U}$, a J-pullback $(X.f, \mathsf{p}_f, \mathsf{Q}(X, f))$ of p along f. A *universe relative to J*, or briefly a *J-relative universe*, is a map p equipped with a J-universe structure.

A *weak universe relative to J* is a map $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ such that for all suitable X, f, there exists some J-pullback of p along f.

A *universe* in $\mathcal{C}$, as defined in [13], is exactly a universe relative to the identity functor $\mathsf{Id}_{\mathcal{C}}$. We will see in Section 5 that universes relative to the Yoneda embedding $\mathsf{y}_{\mathcal{C}} : \mathcal{C} \to \mathsf{PreShv}(\mathcal{C})$ correspond precisely to CwF structures on $\mathcal{C}$.

▶ **Lemma 22** (`isaprop_rel_universe_structure`). *Suppose $J : \mathcal{C} \to \mathcal{D}$ is full and faithful, and $\mathcal{C}$ is univalent. Then for any morphism $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ in $\mathcal{D}$, and object $X : \mathcal{C}$ and $f : J(X) \to \mathsf{U}$, the type of J-pullbacks of p along f is a proposition. Similarly, for any such p, the type of J-universe structures on p is a proposition.*

Proof. The first statement is similar to the argument that pullbacks in univalent categories are unique. The second follows directly from the first. ◀

▶ **Corollary 23** (`weq_relative_universe_weak_relative_universe`). *When $J : \mathcal{C} \to \mathcal{D}$ is fully faithful and $\mathcal{C}$ is univalent, the forgetful function from universes to weak universes relative to J is an equivalence.*

4.2 Transfer constructions

We give three constructions for transferring (weak) relative universes from one functor to another. The first, with all data assumed to be given explicitly, is the most straightforward. However, it does not suffice for transfers to the Rezk completion, since the embedding $\eta_{\mathcal{C}} : \mathcal{C} \to \mathsf{RC}(\mathcal{C})$ is only a weak equivalence, not in general *split* essentially surjective. The second and third constructions are therefore adaptations of the first to require only essential surjectivity.

▶ **Problem 24.** *Given a square of functors commuting up to natural isomorphism, as in*

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{J} & \mathcal{D} \\ {\scriptstyle R}\downarrow & \swarrow_{\alpha} & \downarrow{\scriptstyle S} \\ \mathcal{C}' & \xrightarrow[J']{} & \mathcal{D}', \end{array}$$

such that

- *S preserves pullbacks,*
- *S is split full, and*
- *R is split essentially surjective,*

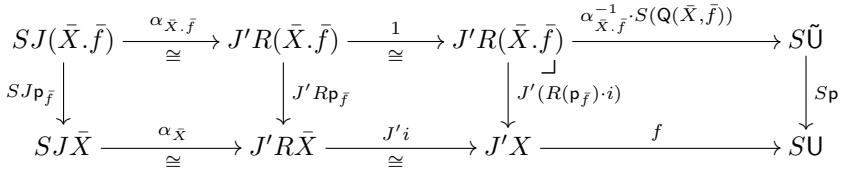
and given a J-relative universe structure on a map $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ in $\mathcal{D}$, to construct a J'-universe structure on $S(\mathsf{p})$ in $\mathcal{D}'$.

▶ **Construction 25** (for Problem 24; `rel_universe_structure_induced_with_ess_split`). Given X in $\mathcal{C}'$ and $f : J'(X) \to S\mathsf{U}$, we need to construct a J'-pullback of $S(\mathsf{p})$ along f.

Split essential surjectivity of R gives some $\bar{X} : \mathcal{C}$ and isomorphism $i : R(\bar{X}) \cong X$. We therefore have $\alpha_{\bar{X}} \cdot J'i \cdot f : SJ\bar{X} \to S\mathsf{U}$; so split fullness of S gives us some $\bar{f} : J\bar{X} \to \mathsf{U}$ with $S\bar{f} = \alpha_{\bar{X}} \cdot J'i \cdot f$. Taking the given J-pullback of p along $\bar{f}$ and mapping it forward under S, we get a pullback square in $\mathcal{D}'$:

$$\begin{array}{ccccccc} SJ(\bar{X}.\bar{f}) & & \xrightarrow{\qquad\qquad S(\mathsf{Q}(\bar{X},\bar{f}))\qquad\qquad} & & & & S\tilde{\mathsf{U}} \\ {\scriptstyle SJ\mathsf{p}_{\bar{f}}}\downarrow & \lrcorner & & & & & \downarrow{\scriptstyle S\mathsf{p}} \\ SJ\bar{X} & \xrightarrow{\alpha_{\bar{X}}} & J'R\bar{X} & \xrightarrow{J'i} & J'X & \xrightarrow{f} & S\mathsf{U} \end{array}$$

The maps $R(\mathsf{p}_{\bar{f}}) \cdot i : R(\bar{X}.\bar{f}) \to X$ and $\alpha^{-1}_{\bar{X}.\bar{f}} \cdot S(\mathsf{Q}(\bar{X},\bar{f})) : J'R(\bar{X}.\bar{f}) \to S\tilde{\mathsf{U}}$ now give the desired J'-pullback of $S\mathsf{p}$ along f, since the square they give – the right-hand square below – is isomorphic to the pullback obtained above.



▶ **Problem 26.** *Given a square of functors as in Problem 24, such that*

- *S preserves pullbacks,*
- *S is full,*
- *R is essentially surjective,*
- *$\mathcal{C}'$ is univalent,*
- *J' is fully faithful,*

and given a J-relative universe structure on a map $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ in $\mathcal{D}$, to construct a J'-universe structure on $S(\mathsf{p})$ in $\mathcal{D}'$.

▶ **Construction 27** (for Problem 26; `rel_universe_structure_induced_with_ess_surj`)**.** Mostly the same as Construction 25. The only problematic steps are finding $(\bar{X}, i)$ and $\bar{f}$ as above, since the hypotheses which provided them there have now been weakened to existence properties. However, our new hypotheses that $\mathcal{C}'$ is univalent and J' fully faithful allow us to apply Lemma 22 to see that the goal of a J'-pullback is a proposition; so the existence properties do in fact suffice. ◀

▶ **Lemma 28** (`is_universe_transfer`)**.** *Suppose given a square of functors as in Problem 24, such that*

- *S preserves pullbacks,*
- *S is full,*
- *R is essentially surjective.*

If $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ in $\mathcal{D}$ is a weak J-relative universe in $\mathcal{C}$, then $S(\mathsf{p})$ is a weak J'-relative universe in $\mathcal{D}'$.

Proof. Again, mostly the same as Construction 25. Now our goal is just to show, for each suitable X, f, that there exists some J'-pullback of $S\mathsf{p}$ along f. This is by construction a proposition; so, again, we can obtain objects from our existence hypotheses whenever required. ◀

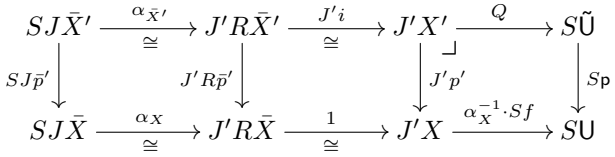
▶ **Lemma 29** (`isweq_is_universe_transfer`)**.** *Given functors satisfying the hypotheses of Lemma 28, if additionally R is full and S is faithful, then a map $\mathsf{p} : \tilde{\mathsf{U}} \to \mathsf{U}$ in $\mathcal{D}$ is a weak J-relative universe if and only if $S(\mathsf{p})$ is a weak J'-relative universe.*

Proof. One implication is exactly Lemma 28. For the other, assume that $S(\mathsf{p})$ is a weak J'-relative universe; we must show that p is a weak J-relative universe.

Given $X : \mathcal{C}$ and $f : J(X) \to \mathsf{U}$, we need to show there exists a J-pullback for p along f. Since the goal is just existence, we can take witnesses for existence hypotheses as needed. In particular, we can take some $X' : \mathcal{C}'$, $p' : X' \to RX$ and $Q : J'X' \to S\tilde{\mathsf{U}}$ forming a J-pullback for $S\mathsf{p}$ along $\alpha_X^{-1} \cdot Sf : J'RX \to S\mathsf{U}$.

By essential surjectivity of R, we can find some $\bar{X}' : \mathcal{C}$ and isomorphism $i : R\bar{X}' \cong X'$; and similarly by its fullness, we can take some arrow $\bar{p}' : \bar{X}' \to X$ such that $R(\bar{p}') = i \cdot p'$. Finally, fullness of S gives a map $\bar{Q} : J\bar{X}' \to \tilde{\mathsf{U}}$ with $S\bar{Q} = \alpha_{\bar{X}'} \cdot J'i \cdot Q$.

But now $(\bar{X}', \bar{p}', \bar{Q})$ form a J-pullback for p along f, since under S, the square they give becomes isomorphic to the original J'-pullback square of (X', p', Q), and S reflects limits, as it is full and faithful.



◀

▶ **Problem 30.** *Given a square of functors as in Problem 24, such that*

- *$\mathcal{D}$ and $\mathcal{D}'$ are both univalent,*
- *S is an equivalence, and*
- *R is essentially surjective and full,*

to construct an equivalance between weak J-relative universes and weak J'-relative universes.

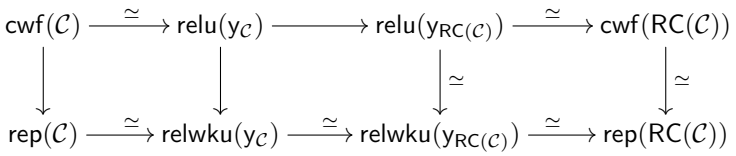
▶ **Construction 31** (for Problem 30; `weq_weak_relative_universe_transfer`)**.** The equivalence S of univalent categories induces an equivalence between morphisms in $\mathcal{D}$ and in $\mathcal{D}'$. Lemma 29 implies that this restricts to an equivalence between weak relative universes as desired. ◀

5 CwF structures, representable maps of presheaves and the Rezk completion

In this section, we show that CwF structures can be seen as relative universes, and hence apply the results of the previous section to transfer CwF structures along weak equivalences of categories: in particular, from a category to its Rezk completion.

We also consider *representable maps of presheaves*, corresponding similarly to relative *weak* universes, and use the results on relative universes to elucidate their relationship to CwF structures.

The resulting transfers and relationships are summed up in the following diagram, whose vertical maps all simply forget chosen structure:



5.1 Representable maps of presheaves

▶ **Definition 32** (`rep_map`)**.** A map $\mathsf{p} : \mathsf{Tm} \to \mathsf{Ty}$ of presheaves on a category $\mathcal{C}$ is *representable* if for each for each $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$, there exists some representation of the fiber of p over A, i.e. some $\pi_A : \Gamma.A \to \Gamma$ and $\mathsf{te}_A \in \mathsf{Tm}(\Gamma.A)$ satisfying the conditions of Definition 1, item 3.[2]

In other words, a representable map of presheaves on $\mathcal{C}$ is just like a CwF structure, except that the representations are merely assumed to exist, not included as chosen data.

[2] Note that in [3], Awodey takes *representable map* to mean what we call a CwF structure, i.e. to include choices of representations of the fibers; cf. [3, §1.1, Algebraic character].

Evidently, the underlying map $\mathsf{p} : \mathsf{Ty} \to \mathsf{Tm}$ of any CwF structure is representable, just by forgetting its chosen representations. This can sometimes be reversed:

▶ **Lemma 33** (`isweq_from_cwf_to_rep`). *If $\mathcal{C}$ is a univalent category, then the forgetful map from CwF structures to representable maps of presheaves on $\mathcal{C}$ is an equivalence. That is, any representable map of presheaves on $\mathcal{C}$ carries a unique choice of representing data.*

Proof. If suffices to show that for a given map $\mathsf{p} : \mathsf{Tm} \to \mathsf{Ty}$, and for any $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$, representing data $(\Gamma.A, \pi_A, \mathsf{te}_A)$ for the fiber is unique if it exists.

Such data is always unique up to isomorphism, for any category $\mathcal{C}$, since pullbacks are unique up to isomorphism and $\mathsf{y}_{\mathcal{C}}$ is full and faithful. But when $\mathcal{C}$ is univalent, this uniqueness up to isomorphism can be translated into literal uniqueness, as required.

(Alternatively, following Constructions 35 and 37 below, we could see this result as essentially a special case of Corollary 23 on universe structures.) ◀

5.2 Transfer of CwF structures and representable maps of presheaves

In order to apply the transfer results of the previous section, we first establish the equivalences between CwF structures on a category $\mathcal{C}$ (resp. representable maps of presheaves) and relative (weak) universes on $\mathsf{y}_{\mathcal{C}}$.

▶ **Problem 34.** *Given a category $\mathcal{C}$, to construct an equivalence between* $\mathsf{cwf}(\mathcal{C})$ *and* $\mathsf{relu}(\mathsf{y}_{\mathcal{C}})$.

▶ **Construction 35** (for Problem 34; `weq_cwf_structure_RelUnivYo`). This is a matter of reassociating components, and replacing two quantifications over elements of a presheaf – once over $\mathsf{Ty}(\Gamma)$, once over $\mathsf{Tm}(\Gamma.A)$ – by quantification over the respective isomorphic sets of natural transformations into Ty and Tm. ◀

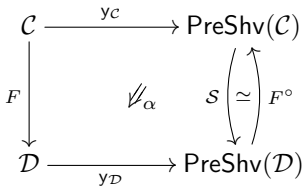
▶ **Problem 36.** *Given a category $\mathcal{C}$, to construct an equivalence between* $\mathsf{rep}(\mathcal{C})$ *and* $\mathsf{relwku}(\mathsf{y}_{\mathcal{C}})$.

▶ **Construction 37** (for Problem 36; `weq_rep_map_weakRelUnivYo`). Similar to Construction 35. ◀

Next, we make use of this to transfer CwF structures and representable maps along weak equivalences, by viewing them as relative (weak) universes and applying the transfer results for those.

▶ **Problem 38.** *Given a weak equivalence $F : \mathcal{C} \to \mathcal{D}$, where $\mathcal{D}$ is univalent, to construct a map* $\mathsf{cwf}(\mathcal{C}) \to \mathsf{cwf}(\mathcal{D})$.

▶ **Construction 39** (for Problem 38; `transfer_cwf_weak_equivalence`). We construct a map $\mathsf{relu}(\mathsf{y}_{\mathcal{C}}) \to \mathsf{relu}(\mathsf{y}_{\mathcal{D}})$ as an instance of Construction 27 and obtain the desired map by composition with the equivalence to CwF structures of Construction 35. Consider the diagram



Here, the functor F° given by precomposition with F^{op} is a weak equivalence between univalent categories, and hence a strong equivalence with inverse $\mathcal{S}$. The isomorphism α

is constructed as follows: Note that fully faithful functors reflect isomorphisms; we apply this for the functor F° of precomposition with F^{op}. It hence suffices to construct a natural isomorphism from $\mathsf{y}_{\mathcal{C}} \cdot \mathcal{S} \cdot F^\circ \simeq \mathsf{y}_{\mathcal{C}}$ to $F \cdot \mathsf{y}_{\mathcal{D}} \cdot F^\circ$. But this is an instance of a general isomorphism: indeed, for any functor $G : \mathcal{A} \to \mathcal{X}$, we have natural transformation from $\mathsf{y}_{\mathcal{A}}$ to $G \cdot \mathsf{y}_{\mathcal{X}} \cdot G^\circ$, and this natural transformation is an isomorphism when G is fully faithful. This ends the construction of the natural isomorphism α. The functor $\mathcal{S}$ preserves pullbacks since it is fully faithful and essentially surjective. The hypotheses of Problem 26 are easily checked, hence Construction 27 applies. ◀

▶ **Problem 40.** *Given a weak equivalence $F : \mathcal{C} \to \mathcal{D}$, to construct an equivalence* $\mathsf{rep}(\mathcal{C}) \simeq \mathsf{rep}(\mathcal{D})$.

▶ **Construction 41** (for Problem 40; `transfer_rep_map_weak_equivalence`)**.** A direct instance of Construction 31. ◀

Putting everything together, we obtain:

▶ **Problem 42.** *For any category $\mathcal{C}$, to construct an equivalence between representable maps on $\mathcal{C}$ and CwF structures on* $\mathsf{RC}(\mathcal{C})$.

▶ **Construction 43** (for Problem 42; `weq_rep_map_cwf_Rezk`)**.** Construction 41, applied to $\eta_{\mathcal{C}}$, gives us an equivalence $\mathsf{rep}(\mathcal{C}) \simeq \mathsf{rep}(\mathsf{RC}(\mathcal{C}))$. On the other hand, Lemma 33 tells us that $\mathsf{cwf}(\mathsf{RC}(\mathcal{C})) \simeq \mathsf{rep}(\mathsf{RC}(\mathcal{C}))$. Composing the first of these with the inverse of the second yields the desired equivalence. ◀

6 Guide to the accompanying formalization

All constructions and theorems of this work have been formalized in the proof assistant Coq, over the `UniMath` library of univalent mathematics [16, 14]. We rely particularly heavily on `UniMath`'s category theory library.

Our formalization can be found at `https://github.com/UniMath/TypeTheory`. The version current at time of writing will remain permanently available under the tag `2017-ALV1`.

The main library will continue development, so naming, organisation, etc. may change from what is presented here. However, the file `Articles/ALV_2017.v` will be maintained to keep the main results of this paper available and locatable over future versions of the library.

For the reader interested in exploring the formalization, we recommend starting with that file, and following backwards to find the details of definitions and constructions. A browsable version is available at `https://unimath.github.io/TypeTheory/coqdoc/master/TypeTheory.Articles.ALV_2017.html`.

The specific material of the present article amounts to about 3700 lines of code in the formalization. Additionally, this development required formalizing a further c. 1500 lines of general background material (mostly on category theory) that had not previously been given.

6.1 Type theory of the formalisation

There are some subtleties in how the type theory of the formalisation relates to what we set out in Section 2.2.

Firstly, the type theory of Coq includes many powerful features; like `UniMath` itself, we deliberately avoid most of these, staying within the fragment described in [14, §1].

Secondly, functional extensionality and univalence are provided by `UniMath` as axioms – that is, as assumptions added to the global context. We use functional extensionality freely, but do not use univalence except where explicitly required, as noted in the text.

Thirdly, in order to acquire resizing principles (not otherwise available in Coq), `UniMath` uses type-in-type, and hence is in principle inconsistent. `UniMath` itself is careful not to use any consequences of type-in-type except for the resulting resizing principles. We are even more restricted: we do not make direct or essential use of the resizing principles. We do depend on them indirectly, since propositional truncation is implemented in `UniMath` using resizing principles. However, our use of truncation is always via the interface corresponding to its standalone axiomatisation, as assumed in Section 2.2.

7 Summary and future work

The above sections complete the construction of the maps and equivalences of types promised in the introduction: in particular,

- equivalence between split type-category structures and CwF structures;
- equivalence between CwF structures and universes relative to the Yoneda embedding, and similarly between representable maps and weak such relative universes;
- transfer of CwF structures and representable maps to the Rezk completion;
- equivalence between CwF structures on a category $\mathcal{C}$ and representable maps of presheaves on its Rezk completion.

There are several natural interesting directions for further work:

- Define *categories* of all the various structures considered here; show that the comparison constructions given here are all moreover functorial, and that our equivalences of types underlie equivalences of categories. Besides its intrinsic interest, this would make more of our constructions meaningful and useful in the classical setting.
- Extend these constructions to give comparisons with other categorical structures considered for similar purposes in the literature: Dybjer's original CwF's; Cartmell's contextual categories/C-systems; comprehension categories; categories with display maps...
- Understand further the transfer of CwF structures along the Rezk completion construction: does the result enjoy an analogous universal property, making it the 'free univalent category with families' on a category $\mathcal{C}$?

Progress in some of these directions may be already found in our formalization, though not included in the present article.

References

1 Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science*, 25:1010–1039, 2015. `arXiv:1303.0584`, `doi:10.1017/S0960129514000486`.

2 Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. Monads need not be endofunctors. *Logical Methods in Computer Science*, 11(1), 2015. `doi:10.2168/LMCS-11(1:3)2015`.

3 Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, pages 1–46, 2016. `arXiv:1406.3219`, `doi:10.1017/S0960129516000268`.

4 John Cartmell. Generalised algebraic theories and contextual categories. *Ph.D. Thesis, Oxford University*, 1978. URL: `http://www.cs.ru.nl/~spitters/Cartmell.pdf`.

5 John Cartmell. Generalised algebraic theories and contextual categories. *Ann. Pure Appl. Logic*, 32:209–243, 1986. `doi:10.1016/0168-0072(86)90053-9`.

6 Peter Dybjer. Internal type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer, 1995. `doi:10.1007/3-540-61780-9_66`.

7 Marcelo Fiore. Discrete generalised polynomial functors, 2012. Slides from talk given at ICALP 2012. URL: http://www.cl.cam.ac.uk/~mpf23/talks/ICALP2012.pdf.

8 Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997.

9 William A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980. Reprint of 1969 article.

10 Andrew M. Pitts. Categorical logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures*, chapter 2, pages 39–128. Oxford University Press, 2000. URL: http://www.oup.co.uk/isbn/0-19-853781-6.

11 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. The Univalent Foundations Program, Institute for Advanced Study, 2013. URL: http://homotopytypetheory.org/book.

12 Benno van den Berg and Richard Garner. Topological and simplicial models of identity types. *ACM Trans. Comput. Logic*, 13(1):3:1–3:44, January 2012. doi:10.1145/2071368.2071371.

13 Vladimir Voevodsky. A C-system defined by a universe category. *Theory Appl. Categ.*, 30(37):1181–1215, 2015. URL: http://www.tac.mta.ca/tac/volumes/30/37/30-37.pdf.

14 Vladimir Voevodsky. An experimental library of formalized mathematics based on the univalent foundations. *Math. Structures Comput. Sci.*, 25(5):1278–1294, 2015. doi:10.1017/S0960129514000577.

15 Vladimir Voevodsky. Subsystems and regular quotients of C-systems. In *Conference on Mathematics and its Applications, (Kuwait City, 2014)*, number 658 in Contemporary Mathematics, pages 127–137, 2016. URL: http://arxiv.org/abs/1406.7413.

16 Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. *UniMath*: Univalent Mathematics. URL: https://github.com/UniMat.

Removing Cycles from Proofs*.

Andrea Aler Tubella[1], Alessio Guglielmi[2], and Benjamin Ralph[3]

1 **IRIF, CNRS et Université Paris Diderot, Paris, France**
`Andrea.Aler@irif.fr`
2 **Department of Computer Science, University of Bath, Bath, UK**
`a.guglielmi@bath.ac.uk`
3 **Department of Computer Science, University of Bath, Bath, UK**
`b.d.ralph@bath.ac.uk`

Abstract

If we track atom occurrences in classical propositional proofs in deep inference, we see that they can form cyclic structures between cuts and identity steps. These cycles are an obstacle to a very natural form of normalisation, that simply unfolds all the contractions in a proof. This mechanism, which we call 'decomposition', has many points of contact with explicit substitutions in lambda calculi. In the presence of cycles, decomposition does not terminate, and this is an obvious drawback if we want to interpret proofs computationally. One way of eliminating cycles is eliminating cuts. However, we could ask ourselves whether it is possible to eliminate cycles independently of (general) cut elimination. This paper shows an efficient way to do so, therefore establishing the independence of decomposition from cut elimination. In other words, cut elimination in propositional logic can be separated into three separate procedures: 1) cycle elimination, 2) unfolding of contractions, 3) elimination of cuts in the linear fragment.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases proof theory, deep inference, proof complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.9

1 Introduction

It is well known, in classical and other logics, that cuts compress proofs. Conversely, eliminating cuts, i.e., normalising, expands proofs, and normalisation can be interpreted computationally. While this is a general phenomenon, there is a special case where the situation is not so clear. In [4], Buss introduces the concept of 'logical flow graph' as a useful tool to analyse the complexity of proofs. A logical flow graph is a graph obtained by tracking subformulae in a proof, and the topological information that it exposes can be directly connected to the complexity of the proof. In [5], Buss notes that it is possible to form cycles in the flow graphs of classical logic proofs, where a cycle is essentially a loop involving cuts and identity axioms. In that paper, Buss states the problem of determining whether using cuts in cycles helps to compress a proof significantly, or not, and he offers examples where the cycles can be eliminated at no cost. In [6], Carbone proves that n cycles can be eliminated from a classical propositional proof of k lines to give an acyclic proof of $\mathcal{O}(k^{n+1})$ lines. All those results apply to the sequent calculus. This paper contributes some results concerning essentially the same problem for deep-inference classical propositional proofs [1] and provides a procedure that eliminates cycles.

* This research has been supported by EPSRC Project EP/K018868/1 *Efficient and Natural Proof Systems* and ANR project FISP ANR-15-CE25-0014-01

Let us first introduce the wider context of the present work. Normalisation in deep inference allows a finer control over complexity than Gentzen's theory does, in particular, because it separates two composition mechanisms that in the normalisation of the sequent calculus are conflated: cut and contraction. The notion of logical flow graph becomes much simpler, because, in deep inference, we can reduce all the structural rules to their atomic variants. These simplified flow graphs are called 'atomic flows' [8, 10]. It turns out that we can control normalisation directly from atomic flows because every operation on the graphs can be mimicked by a corresponding operation on inference steps of minimal (atomic) granularity. In deep inference, we have atomic cuts, atomic identities, atomic contractions and atomic cocontractions; inference steps of these kinds can all be moved and normalised upon separately, which is not possible in Gentzen theory. The finer control on complexity that we obtain this way leads to the surprising result that eliminating cuts in propositional logic only has a quasipolynomial cost (as opposed to exponential in Gentzen's theory) [2, 12]. This development depends, crucially, on the use of contractions as a sharing mechanism in a structure that we call a 'sausage'. It is an open problem whether sausages can be, in turn, removed at polynomial cost or not [7]. In other words, in deep inference, part of the complexity that in Gentzen's theory is controlled by cuts has been isolated into contractions.

What about cycles in deep inference? Can they (and their cuts) be eliminated at a lower cost than general cut elimination? This paper makes some progress towards answering that question, in the first place by providing a cycle elimination procedure (independent from general cut elimination) and secondarily by confining the creation of most proof complexity in one specific step of the procedure. As one could expect, again, some of the complexity that was controlled by cuts is shifted to contractions: our procedure produces sausages. However, what is totally unexpected is the way sausages are created. This happens only via normalising through associativity steps, i.e., complexity is created in inference steps that have no logical content in terms of deduction. Why is it so? Is it just an artefact of our procedure? At present, we are unable to devise a procedure that avoids the problem, but we also are unable to design proofs with cycles where the use of the offending associativity is crucial. In other words, it seems possible to enhance our procedure in such a way that some preprocessing of the given proof would avoid the creation of sausages.

Apart from the progress in dealing with proof complexity, cycle elimination helps normalisation theory in general, essentially because it provides a simple induction measure. The experience of a decade of work with atomic flows (and almost three decades with logical flow graphs) shows that paths in proofs play a crucial role in understanding where the pieces of proofs move during normalisation. Typically, subproofs move along paths; for example, contractions move along atomic flows, in a process called 'decomposition' in the deep inference literature. The absence of cycles allows us to use the length of paths as a straightforward induction measure for decomposition. Therefore we expect applications of this research in computational interpretations. A further benefit should be in the development of a proof semantics that takes into consideration proof complexity. More generally, this result fits in a wider ongoing research on separating the various compression mechanisms of proofs. For example, and thanks to this result, cut elimination in propositional logic can be separated into three separate procedures: 1) cycle elimination, 2) decomposition, 3) elimination of cuts in the linear fragment (a process called 'splitting' in the linear logic literature). Finally, we note that the technique employed in this paper is not restricted to classical logic, and, in fact, can be generalised to a wide class of logics [13].

As is well known, the issues of compression and circularity pop up everywhere in computational logic, so, at a superficial level, our work could be deemed to have connections to

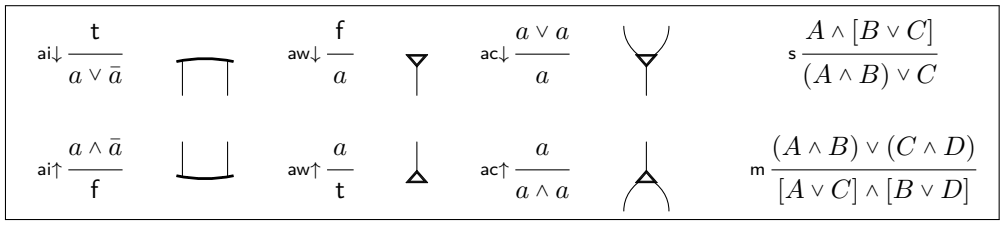


Figure 1 The six structural rules and two logical rules of SKS. The structural rules are shown with their respective atomic flow vertices [3, 11].

several other investigations. On the other hand, it seems that our cycle-elimination procedure only makes sense in proof systems where the full descriptive power of atomic flows can be exploited. This basically means that we need to work on fully localised proof systems, i.e., proof systems where the cost of checking a rule instance is bounded by a constant. This is a feature that (to the best of our knowledge) is only achievable in deep inference.

There is no way that this paper can be made self-contained because of the limitations of the conference format. Luckily, good papers exist on the fundamentals of deep inference. On the other hand, we made some effort towards making the paper self-explanatory regarding atomic flows. In other words, while the reader who does not know deep inference needs to refer to the cited literature, there is no need to study atomic flows elsewhere.

2 Preliminaries on Deep Inference and Atomic Flows

In this paper, we will be working in the deep inference system SKS for classical propositional logic, using the formalism open deduction [9]. We will assume a working knowledge of this proof system, a full exposition of which can be found in [7]. The structural and logical rules of SKS can be found in Figure 1. We do not include the equality (associativity and commutativity of connectives) and unit rules. We will sometimes use the non-atomic versions of the six structural rules (denoted without the 'a'). Each non-atomic structural rule ρ can be thought of as abbreviating a derivation using the rules $\mathsf{a}\rho, \mathsf{s}$ and m [3].

2.1 Atomic Flows

An *(atomic) flow* is a geometric invariant of an SKS derivation that follows the occurrences of atoms. They can be seen as composite diagrams that are freely generated from a set of six elementary diagrams, or as labelled directed graphs, where the six possible labels for the vertices are given in Figure 1. We can associate an atomic flow to every derivation in SKS in a natural way: every edge follows the occurrence of an atom in the derivation, and each vertex label corresponds to the occurrence of a structural rule where atoms are created or destroyed ($\mathsf{ai}{\downarrow}, \mathsf{ai}{\uparrow}, \mathsf{aw}{\downarrow}, \mathsf{aw}{\uparrow}$), or duplicated ($\mathsf{ac}{\downarrow}, \mathsf{ac}{\uparrow}$). The units f and t are not represented in the flow. See Figure 2 for examples of SKS derivations and their respective flows.

Technically, there are some polarity restrictions on the construction of the flows to guarantee that for every flow there is an associated SKS derivation. However, an intuitive understanding of the flows is sufficient to follow the graphical representation of the reduction rules and the measure presented below, and this is what we are seeking to provide. Again, the interested reader is invited to refer to [7] for further technical details on the definition of atomic flows.

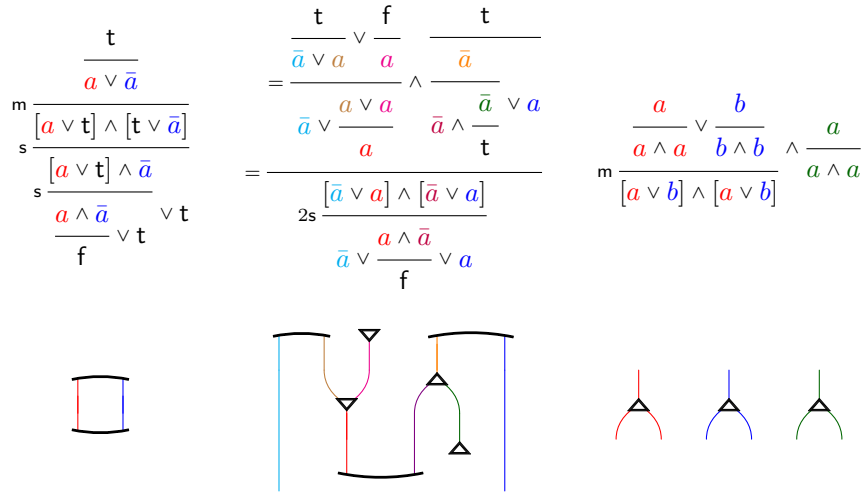


Figure 2 Three examples of SKS derivations and the atomic flows associated to them.

2.2 Decomposition

One major use of atomic flows is to better understand normalisation in deep inference [7, 8]. In particular, we can use atomic flows to describe the aspect of normalisation that deals with (co)contractions and (co)weakenings, preliminary to cut elimination. This stage of normalisation is called *decomposition* [13].

▶ **Definition 1.** An SKS derivation ϕ from A to B can be *decomposed* if we can convert it to the following form:

$$\begin{array}{ccc} A & & A \\ \phi \,\|\, \mathsf{SKS} & \longrightarrow & \|\, \mathsf{aw}\uparrow \\ B & & A_1 \\ & & \|\, \mathsf{ac}\uparrow \\ & & A_2 \\ & & \|\, \{\mathsf{s},\mathsf{m},\mathsf{ai}\uparrow,\mathsf{ai}\downarrow\} \\ & & A_3 \\ & & \|\, \mathsf{ac}\downarrow \\ & & A_4 \\ & & \|\, \mathsf{aw}\downarrow \\ & & B \end{array}$$

We will first show that every acyclic proof can be decomposed, a result first shown in [8], and then extend the result to proofs containing cycles.

2.3 The rewriting system C

▶ **Definition 2.** A *reduction rule* r is a pair (ϕ', ψ') where ϕ' and ψ' are derivations in SKS with the same premise and conclusion. We write $r : \phi' \to \psi'$.

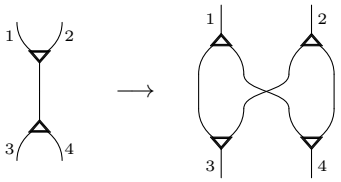
For every reduction rule $r : \phi' \to \psi'$ we define the reduction $\to_r$ such that $\phi \to_r \psi$ if and only if ψ' is a subderivation of ϕ and ψ is obtained from ϕ by replacing ϕ' by ψ'.

We call a finite set R of reduction rules a *rewriting system*. Given a set S of derivations, we say that rewriting system R is *terminating on* S if there is no infinite chain $\phi \to_{r_1} \phi_1 \to_{r_2} \dots$ with $r_i \in R$ for any $\phi \in S$.

▶ **Definition 3.** We define the following reduction rules for SKS:

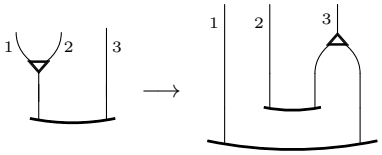
$\mathsf{ac}{\downarrow} - \mathsf{ac}{\uparrow}$:

$$\dfrac{\dfrac{a \vee a}{a}\,\mathsf{ac}{\downarrow}}{\dfrac{a}{a \wedge a}\,\mathsf{ac}{\uparrow}} \longrightarrow \dfrac{\dfrac{a}{a \wedge a}\,\mathsf{ac}{\uparrow} \vee \dfrac{a}{a \wedge a}\,\mathsf{ac}{\uparrow}}{\dfrac{a \vee a}{a}\,\mathsf{ac}{\downarrow} \wedge \dfrac{a \vee a}{a}\,\mathsf{ac}{\downarrow}}\,\mathsf{m}$$



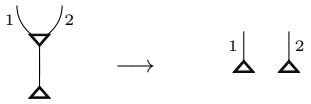
$\mathsf{ac}{\downarrow} - \mathsf{ai}{\uparrow}$:

$$\dfrac{\dfrac{\dfrac{a \vee a}{a}\,\mathsf{ac}{\downarrow} \wedge \bar{a}}{\mathsf{f}}\,\mathsf{ai}{\uparrow}}{\mathsf{f}} \longrightarrow \dfrac{[a \vee a] \vee \dfrac{\bar{a}}{\bar{a} \wedge \bar{a}}\,\mathsf{ac}{\uparrow}}{\dfrac{a \wedge \bar{a}}{\mathsf{f}}\,\mathsf{ai}{\uparrow} \vee \dfrac{a \wedge \bar{a}}{\mathsf{f}}\,\mathsf{ai}{\uparrow}}\,2\mathsf{s}}{=}$$



$\mathsf{ac}{\downarrow} - \mathsf{aw}{\uparrow}$:

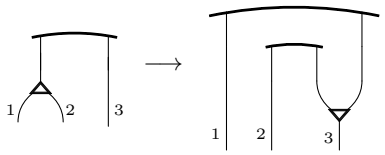
$$\dfrac{\dfrac{\dfrac{a \vee a}{a}\,\mathsf{ac}{\downarrow}}{\mathsf{t}}\,\mathsf{aw}{\uparrow}}{} \longrightarrow \dfrac{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow} \vee \dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow}}{=}$$



And their duals:

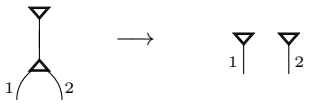
$\mathsf{ai}{\downarrow} - \mathsf{ac}{\uparrow}$:

$$\dfrac{\dfrac{\mathsf{t}}{\dfrac{a}{a \wedge a}\,\mathsf{ac}{\uparrow} \vee \bar{a}}\,\mathsf{ai}{\downarrow}}{} \longrightarrow \dfrac{=\dfrac{\mathsf{t}}{\dfrac{\mathsf{t}}{a \vee \bar{a}}\,\mathsf{ai}{\downarrow} \wedge \dfrac{\mathsf{t}}{a \vee \bar{a}}\,\mathsf{ai}{\downarrow}}\,2\mathsf{s}}{(a \wedge a) \wedge \dfrac{\bar{a} \vee \bar{a}}{\bar{a}}\,\mathsf{ac}{\downarrow}}$$



$\mathsf{aw}{\downarrow} - \mathsf{ac}{\uparrow}$:

$$\dfrac{\dfrac{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}}{a \wedge a}\,\mathsf{ac}{\uparrow}}{} \longrightarrow \dfrac{=\dfrac{\mathsf{f}}{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow} \wedge \dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}}}{}$$



Last, we define the trivial family of reduction rules:

$$\mathsf{ac}{\downarrow} - \rho_H : \dfrac{H\left\{\dfrac{\dfrac{a \vee a}{a}\,\mathsf{ac}{\downarrow}}{}\right\}}{\dfrac{}{H'\{a\}}\,\rho} \longrightarrow \dfrac{\dfrac{H\{a \vee a\}}{H'\left\{\dfrac{a \vee a}{a}\,\mathsf{ac}{\downarrow}\right\}}\,\rho}{}$$

$$\rho_H - \mathsf{ac}{\uparrow} : \dfrac{\dfrac{H'\{a\}}{H\left\{\dfrac{a}{a \wedge a}\,\mathsf{ac}{\uparrow}\right\}}\,\rho}{} \longrightarrow \dfrac{H'\left\{\dfrac{a}{a \wedge a}\,\mathsf{ac}{\uparrow}\right\}}{\dfrac{}{H\{a \wedge a\}}\,\rho}$$

These simply correspond to drawing the (co)contraction node lower (higher) in the atomic flow.

▶ **Definition 4.** We define *rewriting system* C for SKS as the rewriting system given by the reduction rules of Definition 3.

It should be clear that if the rewriting system C terminates for a derivation, we will obtain a derivation with the same premise and conclusion of the following form: all the instances of $\mathsf{ac}\uparrow$ are at the top, followed by a derivation composed only of rules in the set $\{\mathsf{s}, \mathsf{m}, \mathsf{ai}\uparrow, \mathsf{ai}\downarrow, \mathsf{aw}\uparrow, \mathsf{aw}\downarrow\}$ and a bottom phase made up only of $\mathsf{ac}\downarrow$ rules.

2.4 Termination of C

In [8, Theorem 4.22] it is shown that rewriting system C terminates for the set of SKS proofs that do not contain a certain construction, called an ai-cycle. The measure used to prove termination can be easily followed in a flow: it corresponds to the length of a certain type of path.

▶ **Definition 5.** Given an edge ϵ in an atomic flow, we define $\mathsf{up}(\epsilon)$ as the upper vertex it is connected to, and $\mathsf{lo}(\epsilon)$ as the lower vertex it is connected to.

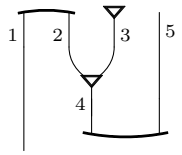
▶ **Definition 6.** Given a sequence of distinct edges $\epsilon_1, \ldots, \epsilon_n$ such that $\mathsf{lo}(\epsilon_i) = \mathsf{up}(\epsilon_{i+1})$ for $1 \leq i < n$, we say that $\epsilon_1, \ldots, \epsilon_n$ is a *path of length n from* $\mathsf{up}(\epsilon_1)$ *to* $\mathsf{lo}(\epsilon_n)$, and that $\epsilon_n, \ldots, \epsilon_1$ is a *path of length n from* $\mathsf{lo}(\epsilon_n)$ *to* $\mathsf{up}(\epsilon_1)$.

Given a sequence of edges $\epsilon_1, \ldots, \epsilon_n$, we say that $\epsilon_1, \ldots, \epsilon_n$ is an ai-*path of length n from vertex* v_1 *to vertex* v_2 if it is a path from v_1 to v_2 or if there exists a vertex v labelled by $\mathsf{ai}\uparrow$ or $\mathsf{ai}\downarrow$ such that $\epsilon_1, \ldots, \epsilon_h$ is an path from v_1 to v, $\epsilon_h \neq \epsilon_{h+1}$, and $\epsilon_{h+1}, \ldots, \epsilon_n$ is an ai-path from v to v_2.

An ai-path of length n is *maximal* if no ai-path containing its edges has length greater than n. An ai-path of length n from v is *maximal* if no ai-path from v containing its edges has length greater than n.

Intuitively, paths correspond to any non-empty sequence of edges from v_1 to v_2 that do not change direction (they either only 'go downwards' or only 'go upwards'). ai-paths are allowed to change direction, but only at ai-vertices: they are zig-zag paths that change direction at ai-nodes.

▶ **Example 7.**



- Some examples of paths in this flow are $(2, 4)$ and (5).
- Some examples of ai-paths in this flow are given by $(1, 2)$ and $(3, 4, 5)$.
- The maximal ai-paths in this flow are $(1, 2, 4, 5)$, $(3, 4, 5)$ and their respective reversals.
- The maximal ai-paths from the $\mathsf{ac}\downarrow$ vertex are $(2, 1)$, (3), and $(4, 5)$.

Informally, the ai-paths from a particular $\mathsf{ac}\downarrow$ vertex correspond to all the paths that the corresponding contraction will take when the reduction rules are applied. Thus, the maximal ai-path length corresponds to the maximal number of other rules a contraction must pass through.

For example, in a derivation whose flow is as in Example 7, when we apply the reduction rules to move the atomic contraction downwards, it will permute with one instance of the rule $\mathsf{ai}\uparrow$.

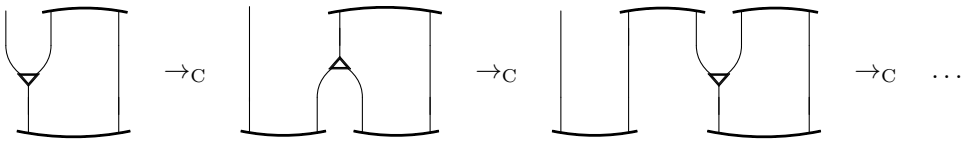


Figure 3 A flow that does not terminate under C.

More precisely, we can assign a *rank* to every contraction and to every cocontraction of a derivation by referring to its flow. The rank of a contraction will be given by the sum of the lengths of the maximal ai-paths starting with the lower edge of its corresponding vertex in the flow. Dually, the rank of a cocontraction will be given by the sum of the lengths of the maximal ai-paths starting with the upper edge of its corresponding vertex in a flow. We will see that the reduction rules of system C reduce the sum of the ranks of the contractions and cocontractions in a derivation, effectively providing a termination measure when these ranks are finite.

▶ **Definition 8.** Given a vertex v labelled with ac↓ in a flow, we define its *rank* as the sum of the lengths of the maximal ai-paths $\epsilon_1, \ldots, \epsilon_n$ from v such that $\mathsf{up}(\epsilon_1) = v$.

Dually, given a vertex v labelled with ac↑ in a flow, we define its *rank* as the sum of the lengths of the maximal ai-paths $\epsilon_1, \ldots, \epsilon_n$ from v such that $\mathsf{lo}(\epsilon_1) = v$.

▶ **Example 9.** The rank of the ac↓ vertex of the flow of example 7 is 2: it corresponds to the length of the ai-path $(4, 5)$.

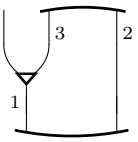
▶ **Definition 10.** Given an occurrence of the rule ac↓ in a derivation ϕ with flow ψ, we define its *rank* as the rank of its corresponding vertex in ψ.

Likewise, we define the rank of an occurrence of the rule ac↑ as the rank of its corresponding vertex.

However, it is possible that (co)contractions have an infinite rank in a derivation: these are precisely those cases when the contraction is in a *cycle*.

▶ **Definition 11.** An ai-path from v to v is called an ai-*cycle*.

▶ **Example 12.** In the following flow, the ai-path $(1, 2, 3)$ is an ai-cycle.



▶ **Definition 13.** We say that a derivation contains an ai-cycle if its atomic flow contains an ai-cycle.

We can easily see that if we repeatedly perform rewrites exclusively on a contraction inside an ai-cycle, such as in Figure 3, then the rewriting procedure will not terminate.

In the absence of such cycles however, the rewriting always terminates. We will briefly outline this result and its proof as presented in [8].

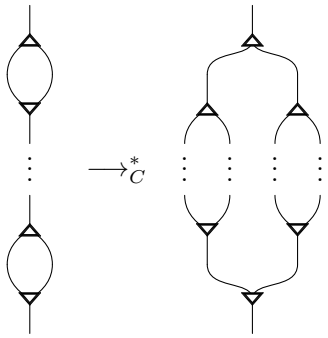


Figure 4 Exponential blow-up caused by sausages.

▶ **Theorem 14.** *The rewriting system* C *is terminating on the set of* ai*-cycle-free derivations.*

Proof. The first observation is that it is clear by inspection of the reduction rules that the rank of (co)contractions not involved in the reduction stays the same.

Given an ai-cycle-free derivation ϕ, we consider the lexicographic order on (r, d). r is the sum of the ranks of the contractions and cocontractions in ϕ, and d is the sum of the number of rules below each contraction and the number of rules above each cocontraction when sequentialising ϕ.

We describe how each application of a reduction of C reduces (r, d):

- Applications of the rules $\mathsf{ac}{\downarrow} - \mathsf{ac}{\uparrow}$, $\mathsf{ac}{\downarrow} - \mathsf{ai}{\downarrow}$ and $\mathsf{ai}{\downarrow} - \mathsf{ac}{\uparrow}$ reduce r in the absence of ai-cycles as is shown in the proof of Theorem 7.2.3 of [11].
- Applications of the rules $\mathsf{ac}{\downarrow} - \mathsf{aw}{\uparrow}$ and $\mathsf{aw}{\downarrow} - \mathsf{ac}{\uparrow}$ reduce r since they remove contractions and cocontractions.
- Applications of the rules $\mathsf{ac}{\downarrow} - \rho_H$ and $\rho_H - \mathsf{ac}{\uparrow}$ trivially maintain r and reduce d. ◀

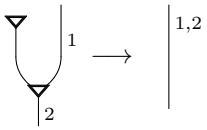
The decomposition procedure may increase the size of a proof exponentially, through the crossings of contractions and cocontractions as shown in Figure 4. We call the contraction-cocontraction pairs on the left *sausages*.

ai-cycles are evidently removed through cut-elimination, since they are caused by the connection of a cut and an introduction. In this paper we will present a local procedure to remove cycles that does not involve cut-elimination, thus proving the independence of decomposition from cut-elimination.

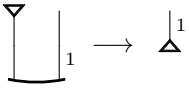
To improve this decomposition result, it can also be shown that (co)weakenings can be permuted to the bottom (top) of a derivation through the following reductions [7].

▶ **Definition 15.** We define the following reduction rules for SKS:

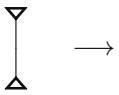
$$\mathsf{aw}{\downarrow} - \mathsf{ac}{\downarrow}: \quad \dfrac{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow} \vee a}{a}\,\mathsf{ac}{\downarrow} \longrightarrow = \dfrac{\mathsf{f} \vee a}{a}$$



$$\mathsf{aw}{\downarrow} - \mathsf{ai}{\uparrow}: \quad \dfrac{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow} \wedge \bar{a}}{\mathsf{f}}\,\mathsf{ai}{\uparrow} \longrightarrow = \dfrac{\mathsf{f} \wedge \mathsf{aw}{\uparrow}\dfrac{\bar{a}}{\mathsf{t}}}{\mathsf{f}}$$

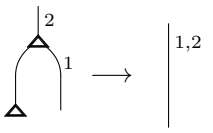


$$\mathsf{aw}{\downarrow} - \mathsf{aw}{\uparrow}: \quad \dfrac{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}}{\mathsf{t}}\,\mathsf{aw}{\uparrow} \longrightarrow = \dfrac{= \dfrac{\mathsf{f}}{\mathsf{f} \wedge [\mathsf{f} \vee \mathsf{t}]}}{\mathsf{t}}\,\mathsf{s}$$

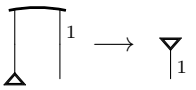


And their duals:

$$\mathsf{ac}{\uparrow} - \mathsf{aw}{\uparrow}: \quad \dfrac{a}{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow} \wedge a}\,\mathsf{ac}{\uparrow} \longrightarrow = \dfrac{a}{\mathsf{t} \wedge a}$$



$$\mathsf{ai}{\downarrow} - \mathsf{aw}{\uparrow}: \quad \dfrac{\mathsf{t}}{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow} \vee \bar{a}}\,\mathsf{ai}{\downarrow} \longrightarrow = \dfrac{\mathsf{t}}{\mathsf{t} \vee \mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{\bar{a}}}$$



And the trivial reductions:

$$\mathsf{aw}{\downarrow} - \rho_H: \quad \dfrac{H\left\{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}\right\}}{H'\{a\}}\,\rho \longrightarrow \dfrac{H\{\mathsf{f}\}}{H'\left\{\dfrac{\mathsf{f}}{a}\,\mathsf{aw}{\downarrow}\right\}}\,\rho$$

$$\rho_H - \mathsf{aw}{\uparrow}: \quad \dfrac{H'\{a\}}{H\left\{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow}\right\}}\,\rho \longrightarrow \dfrac{H'\left\{\dfrac{a}{\mathsf{t}}\,\mathsf{aw}{\uparrow}\right\}}{H\{\mathsf{t}\}}\,\rho$$

▶ **Definition 16.** We define the rewriting system W as the rewriting system given by the reductions in Definition 15.

▶ **Theorem 17.** *The rewriting system* W *is terminating.*

Proof. By observing the corresponding flow reductions, it is easy to see that the non-trivial reductions of W remove edges of atomic flows. Since every application of a non-trivial reduction rule reduces the number of edges of the associated flow to a derivation, and the trivial rules reduce the number of rules below weakenings and above coweakenings, termination is clear. ◀

Note that the reductions of system W do not introduce atomic (co)contractions or medials. Thus, we get the following theorem.

▶ **Theorem 18.** *Given an* SKS *derivation ϕ from A to B not containing* ai*-cycles, we can perform decomposition.*

Proof. By applying system C followed by system W to ϕ. ◀

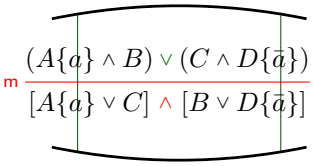
We will now present a local procedure to remove ai-cycles from derivations, effectively showing the independence of decomposition and cut-elimination.

3 The Cycle Elimination Procedure

3.1 Overview

For a cycle to occur in a proof, two edges of an atomic flow that were related by $\vee$ at the top of a connected component have to be connected by $\wedge$ at the bottom of the flow. Therefore, an instance of a rule that changes the main relation between formulae from $\vee$ to $\wedge$ needs to occur, containing the atoms involved in the cycle. In SKS, the only candidate is the medial rule.

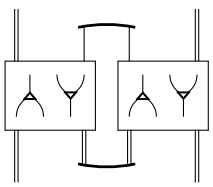
▶ **Definition 19.** A *critical medial* for a cycle is a medial that converts the link between the positive and negative atom from an $\vee$ to a $\wedge$. The picture below shows how this can happen: it depicts the simplest case, the flow containing the cycle can of course be more complicated.



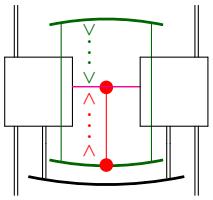
Following this observation, a strategy one can take to remove cycles becomes clear: we can permute the critical instances of the medial rule downwards (or upwards) in a proof. When the corresponding cut is reached, it is 'broken' by the critical medial, and the cycle can then be removed by performing standard deep inference rewriting techniques. It is by no means obvious that this suffices to remove cycles, but we will show that it in fact does.

▶ **Definition 20.** Following the steps below leads to the elimination of ai-cycles in a SKS derivation. An outline of the procedure is given and then the techniques in bold are explained in more detail below. Manipulations of derivations by means of the atomic flow that are not explicitly described follow [11].

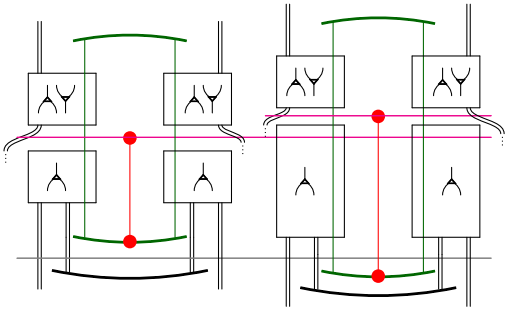
0. **Remove all trivialising units** (this is optional but simplifies step 5).
1. **Normalise every connected component** containing cycles to the following form, where double edges in an atomic flow represent an arbitrary number of edges and boxes represent free compositions of the vertices that they are labelled with:



2. Pick a normalised connected component that contains a cycle. Colour the edges of the cycle green.
3. Starting from the bottom, colour red each conjunction ($\wedge$) connecting two green edges until the critical medial for the cycle. In the same way, colour green each disjunction ($\vee$) connecting two green edges, working from the top of the cycle until the critical medial. We call the trace of $\wedge$-s the $\wedge$-*flow*.



We call the cut below the critical medial the *corresponding cut*.
4. Sequentialise the proof in such a way that the cut corresponding to the bottom critical medial is the top cut, that the cut corresponding the second critical medial from the bottom is the second cut from the top, and so on. **Remove the contractions below each critical medial**.



5. Apply the **transformation along the $\wedge$-flow** to each $\wedge$-flow. Each time apply the transformation to the most internal $\wedge$-flow. This is so no cuts are created and eventually all cycles are removed. The proof so created is broken, but will be fixed in the next phase.
6. Propagate (co)weakenings using the rewriting system W.
7. **Remove all trivialising units.**

3.2 Removing trivialising units

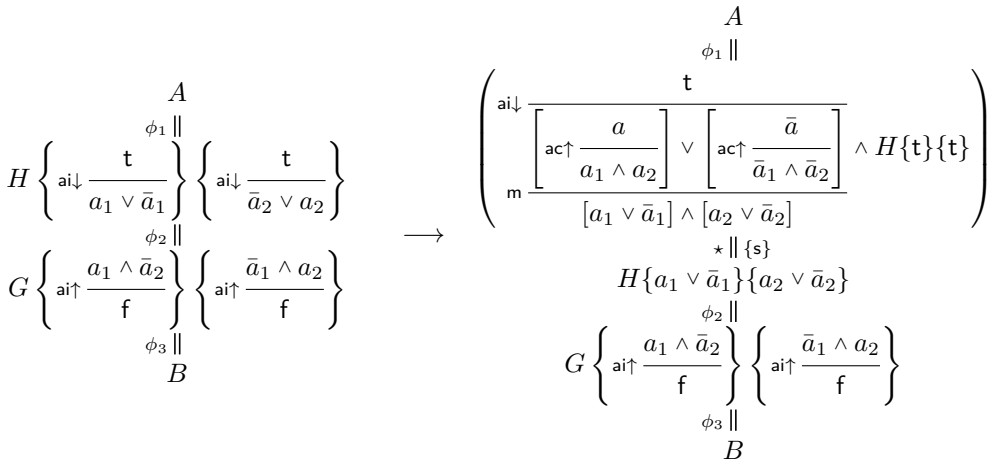
Apply the following transformations, then propagate the (co)-weakenings with rewriting system W:

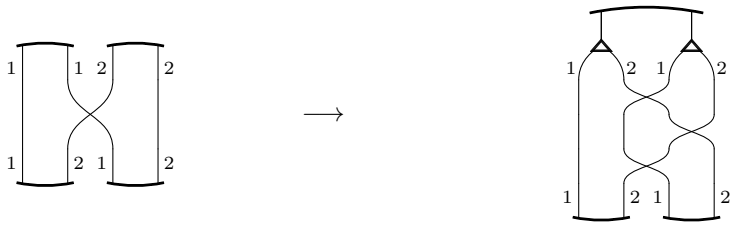
$$\mathsf{f} \wedge A \quad \longrightarrow \quad = \dfrac{\mathsf{f} \wedge \mathsf{w}\uparrow \dfrac{A}{\mathsf{t}}}{\mathsf{aw}\downarrow \dfrac{\mathsf{f}}{A}} \qquad \text{and} \qquad \mathsf{t} \vee A \quad \longrightarrow \quad = \dfrac{\mathsf{w}\uparrow \dfrac{A}{\mathsf{t}}}{\mathsf{t} \vee \mathsf{aw}\downarrow \dfrac{\mathsf{f}}{A}}$$

3.3 Normalise every connected component

To normalise the connected component containing a cycle, we need to collect the identities together. We will show how to do so in the case of two identities; the general case is similar.

Note that the atoms are indexed to ease understanding.

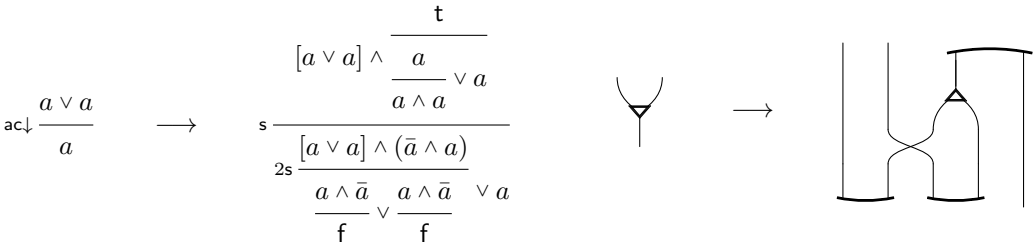




The part of the right-hand derivation labelled with a star is a standard SKS derivation, it can be found in [11, Lemma 2.3.8]. After applying this transformation, the W rewriting system can then be used to get the component into the required form.

3.4 Removing contractions below a critical medial

One could push the contractions through the proof until they are no longer in the wrong place, but, as with any procedure that pushes contractions around, this can have an exponential complexity cost. A simple way to avoid this risk is to convert deviant contractions to cocontractions in the following way, after which the connected component will need to be renormalised, as above.



3.5 Transformation along the ∧-flow

To perform the transformation, we first translate the derivation into sequential form (a dotted line in an inference rule denotes synchronal composition of derivations). We can then

transform along the $\wedge$*-flow* in the following fashion:

$$
\begin{array}{c}
= \dfrac{\chi}{\mathsf{m}\dfrac{(A\{a\} \wedge B) \vee (C \wedge D\{\bar{a}\})}{[A\{a\} \vee C] \wedge [B \vee D\{\bar{a}\}]}} \\
= \dfrac{}{\rho_1 \dfrac{K\{E_1\{a\} \wedge F_1\{\bar{a}\}\}}{\vdots}} \\
\rho_{k-1} \dfrac{}{K\left\{E_k\{a\} \wedge F_k\{\bar{a}\}\right\}} \\
= \dfrac{}{H\left\{\mathsf{ai}{\uparrow}\dfrac{a \wedge \bar{a}}{\mathsf{f}}\right\}} \\
= \dfrac{}{\psi}
\end{array}
\quad \longrightarrow \quad
\begin{array}{c}
= \dfrac{\chi}{K\left\{= \dfrac{(A\{a\} \wedge B) \vee (C \wedge D\{\bar{a}\})}{\left(\left[\dfrac{\mathsf{f}}{A\{a\}} \vee C\right] \wedge \left[\dfrac{\mathsf{f}}{B} \vee D\{\bar{a}\}\right]\right) \vee \left(\left[A\{a\} \vee \dfrac{\mathsf{f}}{C}\right] \wedge \left[B \vee \dfrac{\mathsf{f}}{D\{\bar{a}\}}\right]\right)}\right\}} \\
= \dfrac{}{K\{(E_1\{a\} \wedge F_1\{\bar{a}\}) \vee (E_1\{a\} \wedge F_1\{\bar{a}\})\}} \\
\phi_1 \,\|\, \\
\vdots \\
\phi_{k-1} \,\|\, \\
= \dfrac{K\left\{\left(E_k\{a\} \wedge F_k\{\bar{a}\}\right) \vee \left(E_k\{a\} \wedge F_k\{\bar{a}\}\right)\right\}}{H\left\{\neq \dfrac{\left(a \wedge \mathsf{aw}{\uparrow}\dfrac{\bar{a}}{\mathsf{t}}\right) \vee \left(\mathsf{aw}{\uparrow}\dfrac{a}{\mathsf{t}} \wedge \bar{a}\right)}{\mathsf{f}}\right\}} \\
= \dfrac{}{\psi}
\end{array}
$$

Each inference rule ρ_i on the left yields a derivation ϕ_i as follows. If ρ_i only affects the context $K_i\{\,\}$ or $E_i\{a\} \wedge F_i\{a\}$ then ϕ_i is trivial. Therefore we are left with four non-trivial cases, where P and P represent $E_i\{a\}$ and $E_i\{a\}$, and Q and Q represent $F_i\{a\}$ and $F_i\{a\}$, respectively:

$$
= \frac{(P \wedge Q) \wedge R}{P \wedge (Q \wedge R)} \quad \longrightarrow \quad \mathsf{2s}\frac{[(P \wedge Q) \vee (P \wedge Q)] \wedge \mathsf{c}{\uparrow}\dfrac{R}{R \wedge R}}{= \dfrac{(P \wedge Q) \wedge R}{P \wedge (Q \wedge R)} \vee = \dfrac{(P \wedge Q) \wedge R}{P \wedge (Q \wedge R)}}
$$

$$
= \frac{P \wedge (Q \wedge R)}{(P \wedge Q) \wedge R} \quad \longrightarrow \quad \mathsf{m}\frac{= \dfrac{P \wedge (Q \wedge R)}{(P \wedge Q) \wedge R} \vee = \dfrac{P \wedge (Q \wedge R)}{(P \wedge Q) \wedge R}}{[(P \wedge Q) \vee (P \wedge Q)] \wedge \mathsf{c}{\downarrow}\dfrac{R \vee R}{R}}
$$

$$
\mathsf{m}\frac{(P \wedge Q) \vee (R \wedge S)}{[P \vee R] \wedge [Q \vee S]} \quad \longrightarrow \quad = \frac{[(P \wedge Q) \vee (P \wedge Q)] \vee (R \wedge S)}{\left(\left[P \vee \dfrac{\mathsf{f}}{R}\right] \wedge \left[Q \vee \dfrac{\mathsf{f}}{S}\right]\right) \vee \mathsf{m}\dfrac{(P \wedge Q) \vee (R \wedge S)}{[P \vee R] \wedge [Q \vee S]}}
$$

$$
\mathsf{s}\frac{P \wedge [Q \vee R]}{(P \wedge Q) \vee R} \quad \longrightarrow \quad = \frac{\mathsf{s}\dfrac{P \wedge [Q \vee R]}{(P \wedge Q) \vee R} \vee \mathsf{s}\dfrac{P \wedge [Q \vee R]}{(P \wedge Q) \vee R}}{[(P \wedge Q) \vee (P \wedge Q)] \vee \mathsf{c}{\downarrow}\dfrac{R \vee R}{R}}
$$

The transformed derivation is valid except for the bottom inference rule, labelled $\neq$, which has premise t and conclusion f. Since a and $\bar{a}$ are not involved in contraction steps (we have removed contractions below the critical medial), the weakening steps that generate a and $\bar{a}$ can be propagated down to the invalid inference rule, converting it to the equality $= \dfrac{\mathsf{f} \wedge \mathsf{f}}{\mathsf{f}}$.

3.6 Termination of the Procedure

▶ **Theorem 21.** *The procedure described in Definition 20 terminates, removing all cycles.*

Proof. To see that this procedure to eliminate ai-cycles terminates is straightforward: after each transformation removing a critical medial there is one less cycle, and no new ai-cycles are created in the process. This fact is easy to check: no new connections between existing

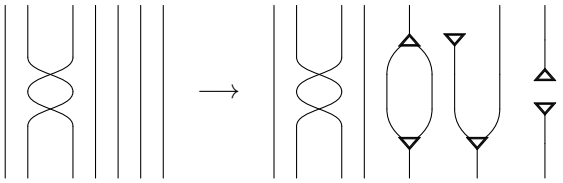


Figure 5 Possible changes to edges after eliminating cycles.

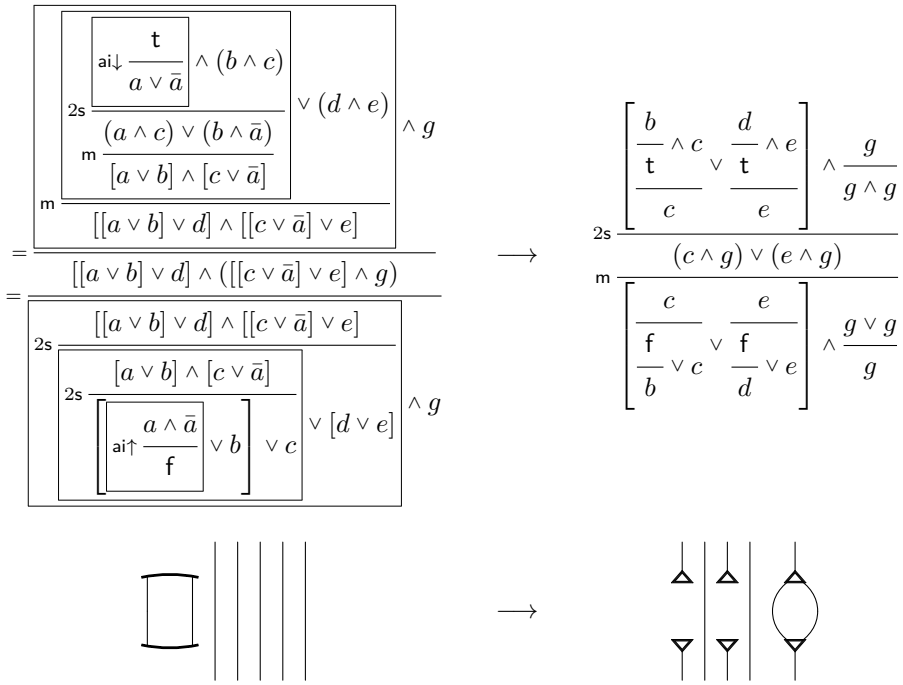


Figure 6 Potential creation of complexity by consecutive associativity rules.

edges and no new cuts are created through this procedure and so the edges connected by a cut-rule after the procedure were already connected by a cut-rule before the procedure. ◀

Thus, from Theorem 21 and Theorem 18, we have that any SKS derivation ϕ from A to B can be decomposed without performing cut elimination.

4 Complexity, Confluence and Open Problems

During the transformation along the $\wedge$ flow, the only possible changes to edges outside the cycle are as follows:

- Edges may be bifurcated and then reconnected, creating a sausage.
- Edges might be joined by another edge which originates from a (co)weakening.
- Edges might be disconnected by weakenings.

Each of these three possibilities is respectively shown in Figure 5 by the three edges to the right of the flow. In the worst case, bifurcation, this incurs a linear cost in the number of

inference rules. Furthermore, in this case sausages are introduced in the flow of the proof, and thus the complexity of the decomposition procedure may be exponentially increased as seen in Figure 4.

Remarkably, what creates the sausage through bifurcation is two opposite instances of the associativity rule: in Figure 6, this happens in a completely redundant section of proof.

This leads to the counter-intuitive conclusion that two proofs with cut, equivalent modulo equality rules, can have an exponential difference in the size of their cut-free proofs when cut elimination involves the above procedure. Clearly, we could avoid the sausage by simply eliminating the associativity rules in the left-hand proof. It is not known whether there are proofs where this cannot be done in a way that does not change the proof in a semantically significant way. An extended version of this paper is being written for journal publication, in which complexity issues will be addressed comprehensively.

Another less startling observation is that the transformation along the $\wedge$-flow is non confluent. This can be seen in the third non-trivial case of converting p_i to ϕ_i, involving medial. On the left, weakenings are introduced, but not on the right. Obviously, there is no reason why this should not be the other way around, and so non-confluence is introduced.

References

1. Paola Bruscoli and Alessio Guglielmi. On the proof complexity of deep inference. *ACM Transactions on Computational Logic (TOCL)*, 10(2):14, 2009. doi:10.1145/1462179.1462186.
2. Paola Bruscoli, Alessio Guglielmi, Tom Gundersen, and Michel Parigot. Quasipolynomial normalisation in deep inference via atomic flows and threshold formulae. *Logical Methods in Computer Science*, 12(1):5:1–30, 2016. doi:10.2168/LMCS-12(2:5)2016.
3. Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250, pages 347–361. Springer, 2001. doi:10.1007/3-540-45653-8_24.
4. Samuel R. Buss. The undecidability of k-provability. *Annals of Pure and Applied Logic*, 53(1):75–102, 1991. doi:10.1016/0168-0072(91)90059-U.
5. Samuel R. Buss. Some remarks on lengths of propositional proofs. *Archive for Mathematical Logic*, 34(6):377–394, 1995. doi:10.1007/BF02391554.
6. Alessandra Carbone. The cost of a cycle is a square. *The Journal of Symbolic Logic*, 67(01):35–60, 2002. doi:10.2178/jsl/1190150028.
7. Anupam Das. On the relative proof complexity of deep inference via atomic flows. *Logical Methods in Computer Science*, 11(1):4:1–27, 2015. doi:10.2168/LMCS-11(1:4)2015.
8. Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1):9:1–36, 2008. doi:10.2168/LMCS-4(1:9)2008.
9. Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A proof calculus which reduces syntactic bureaucracy. In *21st International Conference on Rewriting Techniques and Applications (RTA)*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 135–150. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.RTA.2010.135.
10. Alessio Guglielmi, Tom Gundersen, and Lutz Straßburger. Breaking paths in atomic flows for classical logic. In *Logic in Computer Science (LICS), 2010 25th Annual IEEE Symposium on*, pages 284–293. IEEE, 2010. doi:10.1109/LICS.2010.12.
11. Tom Erik Gundersen. *A general view of normalisation through atomic flows.* Thesis, University of Bath, 2009. URL: https://tel.archives-ouvertes.fr/file/index/docid/509241/filename/thesis.pdf.

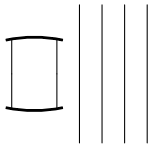
12 Emil Jeřábek. Proof complexity of the cut-free calculus of structures. *Journal of Logic and Computation*, 19(2):323–339, 2008. doi:10.1093/logcom/exn054.

13 Andrea Aler Tubella. *A study of normalisation through subatomic logic*. Thesis, University of Bath, 2017. URL: https://arxiv.org/pdf/1703.10258.pdf.

A An example of cycle elimination

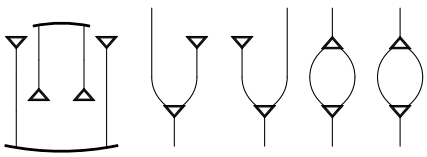
The following proof has a cycle in it, specifically in the atom a. The critical medial is coloured red.

$$= \dfrac{\mathsf{2s}\dfrac{\mathsf{ai}{\downarrow}\dfrac{\mathsf{t}}{a \vee \bar{a}} \wedge (b \wedge c)}{(a \wedge c) \vee (b \wedge \bar{a})} \wedge (d \wedge e) \qquad \mathsf{s}\dfrac{[\bar{a} \vee c] \wedge e}{\bar{a} \vee (c \wedge e)}}{\mathsf{2s}\dfrac{\mathsf{m}\dfrac{[a \vee b] \wedge [c \vee \bar{a}]}{[a \vee b] \wedge d}}{a \vee (b \wedge d)} \wedge \mathsf{s}}}{\mathsf{ai}{\uparrow}\dfrac{a \wedge \bar{a}}{\mathsf{f}} \vee [(b \wedge d) \vee (c \wedge e)]}$$



We perform the cycle removal procedure on this proof. First, we perform the transformation along the $\wedge$-flow:

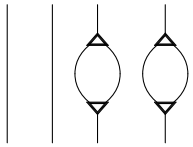
$$= \dfrac{\mathsf{2s}\dfrac{\mathsf{ai}{\downarrow}\dfrac{\mathsf{t}}{a \vee \bar{a}} \wedge (b \wedge c)}{(a \wedge c) \vee (b \wedge \bar{a})} \qquad \wedge\,\mathsf{c}{\downarrow}\dfrac{d \wedge e}{(d \wedge e) \wedge (d \wedge e)}}{\mathsf{2s}\dfrac{\left(\left[\mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{a} \vee b\right] \wedge \left[\mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{c} \vee \bar{a}\right]\right) \vee \left(\left[a \vee \mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{b}\right] \wedge \left[c \vee \mathsf{aw}{\downarrow}\dfrac{\mathsf{f}}{\bar{a}}\right]\right)}{= \dfrac{\mathsf{2s}\dfrac{= \dfrac{\mathsf{s}\dfrac{([a \vee b] \wedge [c \vee \bar{a}]) \wedge (d \wedge e)}{[a \vee b] \wedge d}}{a \vee (b \wedge d)} \wedge \mathsf{s}\dfrac{[c \vee \bar{a}] \wedge e}{\bar{a} \vee (c \wedge e)}}{(a \wedge \bar{a}) \vee [(b \wedge d) \vee (c \wedge e)]} \vee \quad = \dfrac{\mathsf{s}\dfrac{([a \vee b] \wedge [c \vee \bar{a}]) \wedge (d \wedge e)}{[a \vee b] \wedge d}}{a \vee (b \wedge d)} \wedge \mathsf{s}\dfrac{[c \vee \bar{a}] \wedge e}{\bar{a} \vee (c \wedge e)}}{\mathsf{2s}\dfrac{}{(a \wedge \bar{a}) \vee [(b \wedge d) \vee (c \wedge e)]}}}{\neq \dfrac{\left(a \wedge \mathsf{aw}{\uparrow}\dfrac{\bar{a}}{\mathsf{t}}\right) \vee \left(\mathsf{aw}{\uparrow}\dfrac{a}{\mathsf{t}} \wedge \bar{a}\right)}{\mathsf{f}} \vee \mathsf{c}{\downarrow}\dfrac{[(b \wedge d) \vee (c \wedge e)] \vee [(b \wedge d) \vee (c \wedge e)]}{(b \wedge d) \vee (c \wedge e)}}}$$



Note that sausages have been created in the atomic flow, a possible source of complexity.

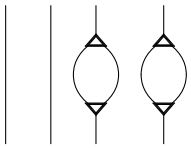
Next we push the weakenings through, using the rewriting system W.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{t}}{\mathsf{t}\vee\mathsf{t}}\wedge(b\wedge c)}{(\mathsf{t}\wedge c)\vee(b\wedge\mathsf{t})}\,{}_{2s} \qquad \dfrac{d\wedge e}{(d\wedge e)\wedge(d\wedge e)}\,{}_{\wedge c\downarrow}}{([\mathsf{f}\vee b]\wedge[\mathsf{f}\vee\mathsf{t}])\vee([\mathsf{t}\vee\mathsf{f}]\wedge[c\vee\mathsf{f}])}\,{}_{2s}}{\dfrac{([\mathsf{f}\vee b]\wedge[\mathsf{f}\vee\mathsf{t}])\wedge(d\wedge e)}{\dfrac{[\mathsf{f}\vee b]\wedge d}{\mathsf{f}\vee(b\wedge d)}\,{}_{s}\wedge\dfrac{[\mathsf{f}\vee\mathsf{t}]\wedge e}{\mathsf{t}\vee(\mathsf{f}\wedge e)}\,{}_{s}}\vee \dfrac{([\mathsf{t}\vee\mathsf{f}]\wedge[c\vee\mathsf{f}])\wedge(d\wedge e)}{\dfrac{[\mathsf{t}\vee\mathsf{f}]\wedge d}{\mathsf{t}\vee(\mathsf{f}\wedge d)}\,{}_{s}\wedge\dfrac{[c\vee\mathsf{f}]\wedge e}{\mathsf{f}\vee(c\wedge e)}\,{}_{s}}}\,{}_{2s}}{\dfrac{(\mathsf{f}\wedge\mathsf{t})\vee[(b\wedge d)\vee(\mathsf{f}\wedge e)]}{\dfrac{(\mathsf{f}\wedge\mathsf{t})\vee(\mathsf{t}\wedge\mathsf{f})}{\mathsf{f}}=\vee\quad \dfrac{[(b\wedge d)\vee(\mathsf{f}\wedge e)]\vee[(\mathsf{f}\wedge d)\vee(c\wedge e)]}{\dfrac{(b\wedge d)\vee(\mathsf{f}\wedge d)}{\dfrac{b\vee\mathsf{f}}{b}=\wedge\,{}_{ac\downarrow}\dfrac{d\vee d}{d}}\,{}_{m}\vee\quad \dfrac{(\mathsf{f}\wedge e)\vee(c\wedge e)}{\dfrac{\mathsf{f}\vee c}{c}=\wedge\,{}_{ac\downarrow}\dfrac{e\vee e}{e}}\,{}_{m}}}$$



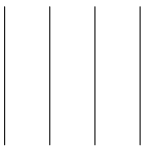
We can simplify this proof using unit equations:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{t}}{\mathsf{t}\vee\mathsf{t}}\wedge(b\wedge c)}{b\vee c}\,{}_{2s} \qquad \dfrac{d\wedge e}{(d\wedge e)\wedge(d\wedge e)}\,{}_{\wedge c\downarrow}}{\dfrac{b\wedge(d\wedge e)}{(b\wedge d)\wedge\dfrac{[\mathsf{t}\vee\mathsf{f}]\wedge e}{\mathsf{t}\vee(\mathsf{f}\wedge e)}\,{}_{s}}\vee \dfrac{c\wedge(d\wedge e)}{\dfrac{[\mathsf{t}\vee\mathsf{f}]\wedge d}{\mathsf{t}\vee(\mathsf{f}\wedge d)}\,{}_{s}\wedge(c\wedge e)}}\,{}_{s}}{\dfrac{(b\wedge d)\vee(\mathsf{f}\wedge e)}{\dfrac{(b\wedge d)\vee(\mathsf{f}\wedge d)}{\dfrac{b\vee\mathsf{f}}{b}=\wedge\,{}_{ac\downarrow}\dfrac{d\vee d}{d}}\,{}_{m}\vee\quad \dfrac{(\mathsf{f}\wedge e)\vee(c\wedge e)}{\dfrac{\mathsf{f}\vee c}{c}=\wedge\,{}_{ac\downarrow}\dfrac{e\vee e}{e}}\,{}_{m}}}\,{}_{2s}$$



We can simplify even further by removing the trivialising units. Note that this eliminates the 'sausages' that appeared after the transformation along the $\wedge$-flow.

$$\dfrac{\dfrac{\dfrac{\mathsf{t}}{\mathsf{t}\vee\mathsf{t}}\wedge(b\wedge c)}{b\vee c}\,{}_{2s}\wedge(d\wedge e)}{(b\wedge d)\vee(c\wedge e)}\,{}_{2s}$$



Query Learning of Derived ω-Tree Languages in Polynomial Time

Dana Angluin[1], Timos Antonopoulos[2], and Dana Fisman[3]

1 **Yale University, New Haven, CT, USA**
2 **Yale University, New Haven, CT, USA**
3 **Ben-Gurion University, Be'er Sheva, Israel**

Abstract

We present the first polynomial time algorithm to learn nontrivial classes of languages of infinite trees. Specifically, our algorithm uses membership and equivalence queries to learn classes of ω-tree languages derived from weak regular ω-word languages in polynomial time. The method is a general polynomial time reduction of learning a class of derived ω-tree languages to learning the underlying class of ω-word languages, for any class of ω-word languages recognized by a deterministic Büchi acceptor. Our reduction, combined with the polynomial time learning algorithm of Maler and Pnueli [15] for the class of weak regular ω-word languages yields the main result. We also show that subset queries that return counterexamples can be implemented in polynomial time using subset queries that return no counterexamples for deterministic or non-deterministic finite word acceptors, and deterministic or non-deterministic Büchi ω-word acceptors.

A previous claim of an algorithm to learn regular ω-trees due to Jayasrirani, Begam and Thomas [11] is unfortunately incorrect, as shown in [4].

1998 ACM Subject Classification F.4.3 Formal Languages, F.1.1 Models of Computation, I.2.6 Learning

Keywords and phrases Learning, queries, infinite trees, derived tree languages, reactive systems

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.10

1 Introduction

Query learning is a framework in which a learning algorithm attempts to identify a target concept using specified types of queries to an oracle (or teacher) about the target concept [2]. For example, if the target concept is a regular language L, a membership query asks whether a string x is a member of L, and is answered either "yes" or "no". An equivalence query asks whether a hypothesis language L' (represented, for example, by a deterministic finite acceptor) is equal to L. In the case of an equivalence query, the answer may be "yes", in which case the learning algorithm has succeeded in exact identification of the target concept, or it may be "no", accompanied by a counterexample, that is, a string x in L but not in L' (or vice versa). The counterexample is a witness that L' is not equal to L.

When L' is not equal to L, there is generally a choice (often an infinity) of possible counterexamples, and we require that the learning algorithm work well regardless of which counterexample is chosen by the teacher. To account for this in terms of quantifying the running time of the learning algorithm, we include a parameter that is the maximum length of any counterexample returned by the teacher at any point in the learning process. In this setting, the L^* algorithm of Angluin [1] learns any regular language L using membership and equivalence queries in time polynomial in the size of the smallest deterministic finite acceptor for L and the length of the longest counterexample chosen by the teacher. There

can be no such polynomial time algorithm using just membership queries or just equivalence queries [3].

The assumption that equivalence queries are available may seem unrealistic. How is a person or a program to judge the equivalence of the target concept to some large, complex, technical specification of a hypothesis? If the hypothesis and the target concept are both deterministic finite acceptors, there is a polynomial time algorithm to test equivalence and return a counterexample in case the answer is negative. Alternatively, if there is an efficient algorithm for exact learnability of a class $\mathbb{C}$ of concepts using membership and equivalence queries, then it is easily transformed to an efficient algorithm that approximately learns concepts from $\mathbb{C}$ using membership queries and a sufficiently large corpus of labeled examples [1]. In this transformation, an equivalence query is answered "yes" if the hypothesis L' agrees with the labels of all the examples in the corpus; otherwise, any exception supplies a counterexample x whose label disagrees with its classification by L'.

Since the publication of L^*, there have been a number of substantial improvements and extensions of the algorithm, as well as novel and unanticipated applications in the analysis, verification and synthesis of programs, protocols and hardware. In a recent CACM review article, Vaandrager [20] explains Model Learning, which takes a black box approach to learning a finite state model of a given hardware or software system using membership queries (implemented by giving the system a sequence of inputs and observing the sequence of outputs) and equivalence queries (implemented using a set of test sequences in which the outputs of the hypothesis are compared with the outputs of the given system.) The learned models may then be analyzed to find discrepancies between a specification and its implementation, or between different implementations. He cites applications in telecommunications, the automotive industry, online conference systems, as well as analyzing botnet protocols, smart card readers, bank card protocols, network protocols and legacy software.

Another application of finite state machine learning algorithms is in the assume-guarantee approach to verifying systems by dividing them into modules that can be verified individually. Cobleigh, Giannakopoulou and Păsareănu [8] first proposed using a learning algorithm to learn a correct and sufficient contextual assumption for the component being verified, and there has since been a great deal of research progress in this area.

If we consider *reactive systems*, that is, systems that maintain an ongoing interaction with their environment (e.g., operating systems, communication protocols, or robotic swarms), the restriction to models specified by finite automata processing finite sequences of inputs is too limiting. Instead, one trajectory of the behavior of a reactive system may be modeled using an infinite word (ω-word), each symbol of which specifies the current state of the system and the environment at a given time. The system itself may be modeled by an ω-automaton, that is, a finite state automaton that processes ω-words. The desired behavior of such a system may be specified by a linear temporal logic formula, that defines the set of ω-words that constitute "good" behaviors of the system.

Researchers have thus sought query learning algorithms for ω-automata that could be used in the settings of model learning or assume-guarantee verification for reactive systems. However, learning ω-automata seems to be a much more challenging problem than learning automata on finite words, in part because the Myhill-Nerode characterization for regular languages (that there is a unique minimum deterministic acceptor that can be constructed using the right congruence classes of the language) does not hold in general for regular ω-languages. The Myhill-Nerode characterization is the basis of the L^* algorithm and its successors.

There is no known polynomial time algorithm using membership and equivalence queries to learn even the whole class $\mathbb{DBW}$ of languages recognized by deterministic Büchi acceptors,

which is a strict subclass of the class of all regular ω-languages. Maler and Pnueli [15] have given a polynomial time algorithm using membership and equivalence queries to learn the *weak regular ω-languages*. This class, denoted $\mathbb{D}\mathbb{w}\mathbb{P}\mathbb{W}$, is the set of languages accepted by deterministic weak parity automata, and is a non-trivial sublass of $\mathbb{D}\mathbb{B}\mathbb{W}$. The class $\mathbb{D}\mathbb{w}\mathbb{P}\mathbb{W}$ does have a Myhill-Nerode property, and the Maler and Pnueli algorithm is a very interesting extension of the L^* approach.

In the context of assume-guarantee verification, Farzan et al. [9] proposed a direct application of L^* to learn the full class of regular ω-languages. Their approach is based on the result of Calbrix, Nivat and Podelski [7] showing that a regular ω-language L can be characterized by the regular language $L_\$$ of finite strings $u\$v$ representing the set of ultimately periodic words $u(v)^\omega$ in L. This establishes that a regular ω-language L is learnable using membership and equivalence queries in time polynomial in the size of the minimal deterministic finite acceptor for $L_\$$. However, the size of this representation may be exponentially larger than its ω-automaton representation. More recently, Angluin and Fisman [5] have given a learning algorithm using membership and equivalence queries for general regular ω-languages represented by families of deterministic finite acceptors, which improves on the $L_\$$ representation, however the running time is not bounded by a polynomial in the representation. Clearly, much more research is needed in the area of query learning of ω-automata.

Despite the difficulties in learning ω-automata (which are used in the analysis of linear temporal logic) in this paper we consider the theoretical question of learning ω-tree automata (which are used in the analysis of branching temporal logic). Though we have not identified any practical application for an algorithm for learning ω-tree languages, we note that our results shed new light on the phenomenon of learning, and we are confident that they will stimulate further work in this challenging area.

Because of the difficulty of the problem, we restrict our attention to ω-tree languages such that all of their paths satisfy a certain temporal logic formula, or equivalently, a property of ω-words. Given an ω-word language L, we use $\mathit{Trees}_d(L)$ to denote the set of all d-ary ω-trees t all of whose paths are in L. The ω-tree language $\mathit{Trees}_d(L)$ is often referred to as the *derived* language of L. In this context, it is natural to ask whether the question of learning a derived ω-tree language $\mathit{Trees}_d(L)$ can be reduced to learning the ω-word language L.

We answer this question affirmatively for the case that L can be recognized by a deterministic Büchi word automaton and learned using membership and equivalence queries. Applying this reduction to the result of Maler and Pnueli on polynomially learning languages in $\mathbb{D}\mathbb{w}\mathbb{P}\mathbb{W}$ we obtain a polynomial learning algorithm for derived languages in $\mathit{Trees}_d(\mathbb{D}\mathbb{w}\mathbb{P}\mathbb{W})$ using membership and equivalence queries. Moreover, any progress on polynomially learning an extended sub-class $\mathbb{C}$ of $\mathbb{D}\mathbb{B}\mathbb{W}$ using membership and equivalence queries can be automatically lifted to learning $\mathit{Trees}_d(\mathbb{C})$.

The framework of the reduction is depicted in Figure 1. An algorithm $\mathbf{A}_{\mathit{Trees}}$ for learning $\mathit{Trees}_d(L)$ uses a learning algorithm $\mathbf{A}$ for L to complete its task. The membership and equivalence queries (MQ and EQ, henceforth) of algorithm $\mathbf{A}_{\mathit{Trees}}$ are answered by respective oracles MQ and EQ for $\mathit{Trees}_d(L)$. Since $\mathbf{A}$ asks membership and equivalence queries about L rather than $\mathit{Trees}_d(L)$, the learner $\mathbf{A}_{\mathit{Trees}}$ needs to find a way to answer these queries. If $\mathbf{A}$ asks a membership query about an ω-word, $\mathbf{A}_{\mathit{Trees}}$ can ask a membership query about an ω-tree all of whose paths are identical to the given ω-word. Since the tree is accepted by $\mathit{Trees}_d(L)$ iff the given word is accepted by L it can pass the answer as is to $\mathbf{A}$. If $\mathbf{A}$ asks an equivalence query, using an acceptor M for an ω-language, then $\mathbf{A}_{\mathit{Trees}}$ can ask an equivalence query using an acceptor M^T that accepts an ω-tree if all its paths are accepted

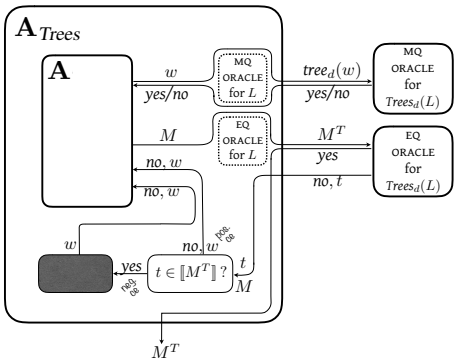


Figure 1 The reduction framework.

by M. If this query is answered positively then $\mathbf{A}_{Trees}$ can output the tree acceptor M^T and halt. The challenge starts when this query is answered negatively.

When the EQ is answered negatively, a counterexample tree t is given. There are two cases to consider. Either this tree is in $Trees_d(L)$ but is rejected by the hypothesis acceptor M^T, in which case t is referred to as a *positive counterexample*; or this tree is not in $Trees_d(L)$ but is accepted by the hypothesis acceptor M^T, in which case t is referred to as a *negative counterexample*. If t is a positive counterexample, since M^T rejects t there must be a path in t which is rejected by M. It is not too dificult to extract that path. The real challenge is dealing with a negative counterexample. This part is grayed out in the figure. In this case the tree t is accepted by M^T yet it is not in $Trees_d(L)$. Thus, all the paths of the tree are accepted by M, yet at least one path is not accepted by L. Since L is not given, it is not clear how we can extract such a path. Since we know that not all paths of t are contained in L, a use of an unrestricted subset query could help us. Unrestricted subset queries (USQ) are queries on the inclusion of a current hypothesis in the unknown language that are answered by "yes" or "no" with an accompanying counterexample in the case the answer is negative.

Since we don't have access to USQs we investigate whether we can obtain such queries given the queries we have. We show that unrestricted subset queries can be simulated by restricted subset queries. Restricted subset queries (RSQ) on ω-words are subset queries that are *not* accompanied by counterexamples. This essentially means that there is a way to construct a desired counterexample without it being given. To discharge the use of restricted subset queries (as the learner is not provided such queries either) we investigate the relation between subsets of ω-words and ω-trees. Finally, we show that the desired subset queries on ω-words can be given to the ω-tree learning algorithm by means of ω-tree membership queries. From these we can construct a series of procedures to implement the grayed area.

The subsequent sections contain definitions of ω-words, ω-trees and automata processing them, derived ω-tree languages, the problem of learning classes of ω-word and ω-tree languages, preliminary results, the algorithm for the main reduction, and discussion. Many proof details are omitted because of space limitations.

2 Definitions

2.1 Words and trees

(For more details see Grädel, Thomas and Wilke [10], Perrin and Pin [18], and Löding [14].) Let Σ be a fixed finite alphabet of symbols. The set of all finite words over Σ is denoted Σ^*.

The empty word is denoted ε, and the length of a finite word x is denoted $|x|$. Σ^+ is the set of all nonempty finite words over Σ, and for a nonnegative integer k, Σ^k is the set of all finite words over Σ of length equal to k. A finite word language is a subset of Σ^*.

An ω-word over Σ is an infinite sequence $w = \sigma_1\sigma_2\sigma_3\cdots$ where each $\sigma_i \in \Sigma$. The set of all ω-words over Σ is denoted Σ^ω. An ω-word language is a subset of Σ^ω. The ω-regular expressions are analogous to finite regular expressions, with the added operation S^ω, where S is a set of finite words, and the restriction that concatenation may combine two sets of finite words, or a set of finite words and a set of ω-words. The set S^ω is the set of all ω-words $s_1s_2\cdots$ such that for each i, $s_i \in S$ and $s_i \neq \varepsilon$. For example, $(a+b)^*(a)^\omega$ is the set of all ω-words over $\{a, b\}$ that contain finitely many occurrences of b.

If $S \subseteq \Sigma^*$, n is a nonnegative integer and $u \in \Sigma^*$, we define the *length and prefix restricted* version of S by $S[n, u] = S \cap \Sigma^n \cap (u \cdot \Sigma^*)$. This is the set of all words in S of length n that begin with the prefix u. We also define the *length restricted* version of S by $S[n] = S[n, \varepsilon]$, that is, the set of all words in S of length n.

Let d be a positive integer. We consider T_d, the unlabeled complete d-ary ω-tree whose directions are specified by $D = \{1, \ldots, d\}$. The *nodes* of T_d are the elements of D^*. The *root* of T_d is the node ε, and the *children* of node v are $v \cdot i$ for $i \in D$. An *infinite path* π in T_d is a sequence $x_0, x_1, x_2, \ldots$ of nodes of T_d such that x_0 is the root and for all nonnegative integers n, x_{n+1} is a child of x_n. An infinite path in T_d corresponds to an ω-word over D giving the sequence of directions traversed by the path starting at the root.

A labeled d-ary ω-tree (or just *ω-tree*) is given by a mapping $t : D^+ \to \Sigma$ that assigns a symbol in Σ to each non-root node of T_d. We may think of t as assigning the symbol $t(v \cdot i)$ to the tree edge from node v to its child node $v \cdot i$. The set of all labeled d-ary ω-trees is denoted T_d^Σ. An ω-tree language is a subset of T_d^Σ. If $\pi = x_0, x_1, x_2, \ldots$ is an infinite path of T_d, then we define $t(\pi)$ to be the ω-word $t(x_1), t(x_2), \ldots$ consisting of the sequence of labels of the non-root nodes of π in t. (Recall that t does not label the root node.)

2.2 Automata on words

A *finite state word automaton* is given by a tuple $M = (Q, q_0, \delta)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma \to 2^Q$ is the (nondeterministic) transition function. The automaton is *deterministic* if $\delta(q, \sigma)$ contains at most one state for every $(q, \sigma) \in Q \times \Sigma$, and *complete* if $\delta(q, \sigma)$ contains at least one state for every $(q, \sigma) \in Q \times \Sigma$. For a complete deterministic automaton we extend δ to map $Q \times \Sigma^*$ to Q in the usual way.

Let $x = \sigma_1\sigma_2\cdots\sigma_k$ be a finite word, where each $\sigma_n \in \Sigma$. A *run* of M on x is a sequence of $k+1$ states $r_0, r_1, \ldots, r_k$ such that $r_0 = q_0$ is the initial state and $r_n \in \delta(r_{n-1}, \sigma_n)$ for integers $1 \leq n \leq k$. Let $w = \sigma_1\sigma_2\cdots$ be an ω-word, where each $\sigma_n \in \Sigma$. A *run* of M on w is an infinite sequence of states $r_0, r_1, r_2, \ldots$ such that $r_0 = q_0$ is the initial state and $r_n \in \delta(r_{n-1}, \sigma_n)$ for all positive integers n.

A *nondeterministic finite acceptor* is given by $M = (Q, q_0, \delta, F)$, where (Q, q_0, δ) is a finite state word automaton, and the new component $F \subseteq Q$ is the set of accepting states. M is a *deterministic finite acceptor* if δ is deterministic. Let M be a nondeterministic finite acceptor and $x \in \Sigma^*$ a finite word of length n. M *accepts* x iff there is a run $r_0, r_1, \ldots, r_n$ of M on x such that $r_n \in F$. The language *recognized* by M is the set of all finite words accepted by M, denoted by $[\![M]\!]$. The class of all finite word languages recognized by deterministic finite acceptors is denoted by $\mathbb{DFW}$, and by nondeterministic finite acceptors, $\mathbb{NFW}$. These representations are equally expressive, that is, $\mathbb{NFW} = \mathbb{DFW}$.

Turning to finite state word automata processing ω-words, a variety of different acceptance criteria have been considered. Such an acceptor is given by a tuple $M = (Q, q_0, \delta, \alpha)$, where

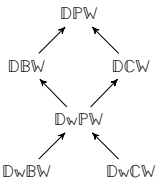


Figure 2 Expressiveness hierarchy of ω-acceptors [21, 16].

(Q, q_0, δ) is a finite state word automaton and α specifies a mapping from 2^Q to $\{0, 1\}$ which gives the criterion of acceptance for an ω-word w.

Given an ω-word w and a run $r = r_0, r_1, \ldots$ of M on w, we consider the set $Inf(r)$ of all states $q \in Q$ such that $r_n = q$ for infinitely many indices n. The acceptor M *accepts* the ω-word w iff there exists a run r of M on w such that $\alpha(Inf(r)) = 1$. That is, M accepts w iff there exists a run of M on w such that the set of states visited infinitely often in the run satisfies the acceptance criterion α. The language *recognized* by M is the set of all ω-words accepted by M, denoted $[\![M]\!]$.

For a Büchi acceptor, the acceptance criterion α is specified by giving a set $F \subseteq Q$ of accepting states and defining $\alpha(S) = 1$ iff $S \cap F \neq \emptyset$. In words, a Büchi acceptor M accepts w if and only if there exists a run r of M on w such that at least one accepting state is visited infinitely often in r. For a co-Büchi acceptor, the acceptance criterion α is specified by giving a set $F \subseteq Q$ of rejecting states and defining $\alpha(S) = 1$ iff $S \cap F = \emptyset$. For a parity acceptor, α is specified by giving a function c mapping Q to an interval of integers $[i, j]$, (called *colors* or *priorities*) and defining $\alpha(S) = 1$ iff the minimum integer in $c(S)$ is even.

A *parity* automaton is said to be *weak* if no two strongly connected states have distinct colors, i.e., if looking at the partition of its states to maximal strongly connected components (MSCCs) all states of an MSCC have the same color. Clearly every weak parity automaton can be colored with only two colors, one even and one odd, in which case the colors are often referred to as *accepting* or *rejecting*. It follows that a weak parity automaton can be regarded as either a Büchi or a coBüchi automaton. If in addition no rejecting MSCC is reachable from an accepting MSCC, the acceptor is said to be *weak Büchi*. Likewise, a weak parity acceptor where no accepting MSCC is reachable from a rejecting MSCC, is said to be *weak coBüchi* acceptor.

The classes of languages of ω-words recognized by these kinds of acceptors will be denoted by three/four-letter acronyms, with $\mathbb{N}$ or $\mathbb{D}$ (for nondeterministic or deterministic), $\mathbb{B}$, $\mathbb{C}$, $\mathbb{P}$, $\mathbb{wB}$, $\mathbb{wC}$ or $\mathbb{wP}$ (for Büchi, co-Büchi, parity or their respective weak variants) and then $\mathbb{W}$ (for ω-words). Thus $\mathbb{DwBW}$ is the class of ω-word languages recognized by deterministic weak Büchi word acceptors. Concerning the expressive power of various types of acceptors, previous research has established the following results. The weak variants are strictly less expressive than the non-weak variants. Deterministic parity automata are more expressive than deterministic Büchi and coBüchi automata and the same is true for their weak variants. These results are summarized in Figure 2. In addition, $\mathbb{NBW} = \mathbb{DPW} = \mathbb{NPW}$ and $\mathbb{DwPW} = \mathbb{DCW} \cap \mathbb{DBW}$. The class of *regular ω-languages* is the class $\mathbb{DPW}$, and the class of *weak regular ω-languages* is the class $\mathbb{DwPW}$.

2.3 Automata on trees

Acceptors on d-ary ω-trees are equipped with analogous accepting conditions. Such an acceptor is given by a tuple $M = (Q, q_0, \delta, \alpha)$, where Q is a finite set of states, $q_0 \in Q$ is

the initial state, the transition function δ is a map from Q and d-tuples of symbols to sets of d-tuples of states, that is, $\delta : Q \times \Sigma^d \to 2^{Q^d}$, and the acceptance criterion α specifies a function from 2^Q to $\{0, 1\}$. We may think of the acceptor as running top down from the root of the tree, at each node nondeterministically choosing a permissible d-tuple of states for the d children of the node depending on the state assigned to the node and the d-tuple of symbols on its outgoing edges.

We define a *run* of M on the ω-tree t as a mapping r from the nodes of T_d to Q such that $r(\varepsilon) = q_0$ and for every node x, we have $(r(x \cdot 1), \ldots, r(x \cdot d)) \in \delta(r(x), (t(x \cdot 1), \ldots, t(x \cdot d)))$. That is, the root is assigned the initial state and for every node, the ordered d-tuple of states assigned to its children is permitted by the transition function. The acceptor M *accepts* the ω-tree t iff there exists a run r of M on t such that for every infinite path π, we have $\alpha(\mathit{Inf}(r(\pi))) = 1$. That is, there must be at least one run in which, for every infinite path, the set of states that occur infinitely often on the path satisfies the acceptance criterion α. The ω-tree language *recognized* by M is the set of all ω-trees accepted by M, denoted $[\![M]\!]$.

The specification of the acceptance criterion α is as for ω-word acceptors, yielding Büchi, co-Büchi and parity ω-tree acceptors. If the transition function specifies at most one permissible d-tuple of states for every element of $Q \times \Sigma^d$, then the acceptor is deterministic. The corresponding classes of ω-tree languages are also denoted by three-letter acronyms, where the last letter is $\mathbb{T}$ for ω-trees. For ω-trees, the class of all regular ω-tree languages is $\mathbb{NPT}$ and $\mathbb{NBT}$ is a proper subclass of $\mathbb{NPT}$. For any automaton or acceptor M, we denote the number of its states by $|M|$.

3 Derived ω-tree languages

Given an ω-tree t we define the ω-word language $\mathit{paths}(t)$ consisting of the ω-words labeling its infinite paths. That is, we define

$$\mathit{paths}(t) = \{t(\pi) \mid \pi \text{ is an infinite path in } T_d\}.$$

If L is an ω-word language and d is a positive integer, we define a corresponding language of d-ary ω-trees *derived from* L as follows:

$$\mathit{Trees}_d(L) = \{t \in T_d^\Sigma \mid \mathit{paths}(t) \subseteq L\}.$$

That is, $\mathit{Trees}_d(L)$ consists of all d-ary ω-trees such that every infinite path in the tree is labeled by an element of L. If $\mathbb{C}$ is any class of ω-word languages, $\mathit{Trees}_d(\mathbb{C})$ denotes the class of all ω-tree languages $\mathit{Trees}_d(L)$ such that $L \in \mathbb{C}$.

3.1 Derived tree languages

Not every regular d-ary ω-tree language can be derived in this way from an ω-word language. As an example, consider the language L_a of all binary ω-trees t over $\Sigma = \{a, b\}$ such that there is at least one node labeled with a. An NBT acceptor can recognize L_a by guessing and checking a path that leads to an a. However, if $L_a = \mathit{Trees}_2(L)$ for some ω-word language L, then because there are ω-trees in L_a that have infinite paths labeled exclusively with b, we must have $b^\omega \in L$, so the binary ω-tree labeled exclusively with b would also be in $\mathit{Trees}_2(L)$, a contradiction.

Given an ω-word acceptor $M = (Q, q_0, \delta, \alpha)$, we may construct a related ω-tree acceptor $M^{T,d} = (Q, q_0, \delta^{T,d}, \alpha)$ as follows. For all $q \in Q$ and all $(\sigma_1, \ldots, \sigma_d) \in \Sigma^d$, define

$$\delta^{T,d}(q, (\sigma_1, \ldots, \sigma_d)) = \{(q_1, \ldots, q_d) \mid \forall i \in D, q_i \in \delta(q, \sigma_i)\}.$$

That is, the acceptor $M^{T,d}$ may continue the computation at a child of a node with any state permitted by M, independently chosen. It is tempting to think that $[\![M^{T,d}]\!] = \mathit{Trees}_d([\![M]\!])$, but this may not be true when M is not deterministic.

▶ **Lemma 1.** *Given an ω-word acceptor M, we have that $[\![M^{T,d}]\!] \subseteq \mathit{Trees}_d([\![M]\!])$ with equality if M is deterministic.*

Boker et al. [6] give the following example to show that the containment asserted in Lemma 1 may be proper if M is not deterministic. The ω-language L specified by $(a+b)^*b^\omega$ can be recognized by the nondeterministic Büchi acceptor M with two states, q_0 and q_1, transition function $\delta(q_0, a) = \{q_0\}$, $\delta(q_0, b) = \{q_0, q_1\}$, $\delta(q_1, b) = \{q_1\}$, and accepting state set $\{q_1\}$. Let $d = 2$, specifying binary trees with directions $\{1, 2\}$. Then $M^{T,2}$ is a nondeterministic ω-tree acceptor, but the following example shows $[\![M^{T,2}]\!] \subsetneq \mathit{Trees}_2(L)$. Consider the binary ω-tree t that labels every node in 1^*2 with a and every other non-root node with b. Clearly $t \in \mathit{Trees}_2(L)$ because every infinite path in t has at most one a, but no run of $M^{T,2}$ can satisfy the acceptance criterion on the path 1^ω. Suppose r were an accepting run of $M^{T,2}$ on t. Then for some $n \geq 0$, $r(1^n)$ would have to be equal to q_1. But then such a mapping r would not be a valid run because 1^n2 is labeled by a and $\delta^{T,2}(q_1, (b, a)) = \emptyset$ because $\delta(q_1, a) = \emptyset$.

3.2 Good for trees

This phenomenon motivates the following definition. An ω-word acceptor M is *good for trees* iff for any positive integer d, $[\![M^{T,d}]\!] = \mathit{Trees}_d([\![M]\!])$. Nondeterministic ω-word acceptors that are good for trees are equivalent in expressive power to deterministic ω-word acceptors, as stated by the following result of Boker et al.

▶ **Theorem 2** ([6])**.** *Let L be a regular ω-word language and $d \geq 2$. If $\mathit{Trees}_d(L)$ is recognized by a nondeterministic ω-tree acceptor with acceptance criterion α, then L can be recognized by a deterministic ω-word acceptor with acceptance criterion α.*

This theorem generalizes prior results of Kupferman, Safra and Vardi for Büchi acceptors [13] and Niwiński and Walukiewicz for parity acceptors [17]. One consequence of Theorem 2 is that when $d \geq 2$, nondeterministic ω-word acceptors that are good for trees are not more expressive than the corresponding deterministic ω-word acceptors. Also, for $d \geq 2$, nondeterminism does not increase expressive power over determinism when recognizing ω-tree languages of the form $\mathit{Trees}_d(L)$. To see this, if N is a nondeterministic ω-tree acceptor with acceptance criterion α recognizing $\mathit{Trees}_d(L)$ then there is a deterministic ω-word acceptor M with acceptance criterion α such that $[\![M]\!] = L$, and $M^{T,d}$ is a deterministic ω-tree acceptor with acceptance criterion α that also recognizes $\mathit{Trees}_d(L)$.

However, it is possible that nondeterminism permits acceptors with smaller numbers of states. Kuperberg and Skrzypczak [12] have shown that for an NBT acceptor M recognizing $\mathit{Trees}_d(L)$, there is a DBW acceptor with at most $|M|^2$ states recognizing L, so nondeterminism gives at most a quadratic savings for Büchi tree acceptors that are good for trees. However, they have also shown that the blowup in the case of nondeterministic co-Büchi tree acceptors (and all higher parity conditions) is necessarily exponential in the worst case.

4 Learning tree languages

We address the problem of learning derived ω-tree languages by giving a polynomial time reduction of the problem of learning $\mathit{Trees}_d(\mathbb{C})$ to the problem of learning $\mathbb{C}$. The paradigm

of learning we consider is exact learning with membership queries and equivalence queries. Maler and Pnueli [15] have given a polynomial time algorithm to learn the class of weak regular ω-languages using membership and equivalence queries. Their algorithm and the reduction we give in Theorem 10 prove the following theorem.

▶ **Theorem 3.** *For every positive integer d, there is a polynomial time algorithm to learn $Trees_d(\mathbb{D}\mathbb{W}\mathbb{P}\mathbb{W})$ using membership and equivalence queries.*

4.1 Representing examples

For a learning algorithm, the examples tested by membership queries and the counterexamples returned by equivalence queries need to be finitely represented. For learning regular ω-word languages, it suffices to consider *ultimately periodic ω-words*, that is, words of the form $u(v)^\omega$ for finite words $u \in \Sigma^*$ and $v \in \Sigma^+$. If two regular ω-word languages agree on all the ultimately periodic ω-words, then they are equal. The pair (u, v) of finite words represents the ultimately periodic word $u(v)^\omega$.

The corresponding class of examples in the case of regular ω-tree languages is the class of *regular ω-trees*. These are ω-trees that have a finite number of nonisomorphic complete infinite subtrees. We represent a regular ω-tree t by a *regular ω-tree automaton* $A_t = (Q, q_0, \delta, \tau)$, where (Q, q_0, δ) is a complete deterministic finite state word automaton over the input alphabet $D = \{1, \ldots, d\}$ and τ is an output function that labels each transition with an element of Σ. That is, $\tau : Q \times D \to \Sigma$. The regular ω-tree t represented by such an automaton A_t is defined as follows. For $x \in D^+$, let $i \in D$ be the last symbol of x and let x' be the rest of x, so that $x = x' \cdot i$. Then define $t(x) = \tau(\delta(q_0, x'), i)$, that is, $t(x)$ is the label assigned by τ to the last transition in the unique run of A_t on x.

Rabin [19] proved that if two regular ω-tree languages agree on all the regular ω-trees then they are equal. Thus, ultimately periodic ω-words and regular ω-trees are proper subsets of examples that are nonetheless sufficient to determine the behavior of regular ω-word and ω-tree acceptors on all ω-words and ω-trees, respectively.

4.2 Types of queries for learning

We consider the situation in which a learning algorithm **A** is attempting to learn an initially unknown target language L of ω-words from a known class $\mathbb{C} \subseteq \mathbb{DBW}$. The information that **A** gets about L is in the form of answers to queries of specific types [2]. The learning algorithm will use membership and equivalence queries; in addition, restricted and unrestricted subset queries will be considered in the proof.

In a *membership query about L*, abbreviated MQ, the algorithm **A** specifies an example as a pair of finite words (u, v) and receives the answer "yes" if $u(v)^\omega \in L$ and "no" otherwise. In an *equivalence query about L*, abbreviated EQ, the algorithm **A** specifies a hypothesis language $[\![M]\!]$ as a DBW acceptor M, and receives either the answer "yes" if $L = [\![M]\!]$, or "no" and a counterexample, that is, a pair of finite words (u, v) such that $u(v)^\omega \in (L \oplus [\![M]\!])$, where $B \oplus C$ denotes the symmetric difference of sets B and C.

In a *restricted subset query about L*, abbreviated RSQ, the algorithm **A** specifies a hypothesis language $[\![M]\!]$ as a DBW acceptor M, and receives the answer "yes" if $[\![M]\!] \subseteq L$ and "no" otherwise. An *unrestricted subset query about L*, abbreviated USQ, is like a restricted subset query, except that in addition to the answer of "no", a counterexample (u, v) is provided such that $u(v)^\omega \in ([\![M]\!] \setminus L)$.

A learning algorithm **A** using specific types of queries *exactly learns* a class $\mathbb{C}$ of ω-word languages iff for every $L \in \mathbb{C}$, the algorithm makes a finite number of queries of the specified

Algorithm 1 : *Accepted?*(A_t, M)

Require: $A_t = (Q_1, q_{0,1}, \delta_1, \tau_1)$ representing t;
$M = (Q_2, q_{0,2}, \delta_2, F_2)$, a complete DBW acceptor
Ensure: Return "yes" if $M^{T,d}$ accepts t
else return "no" and (u, v) with $u(v)^\omega \in (paths(t) \setminus [\![M]\!])$.

```
let Q = Q_1 × Q_2
let q_0 = (q_{0,1}, q_{0,2})
for all (q_1, q_2) ∈ Q and i ∈ D do
    let δ((q_1, q_2), i) = (δ_1(q_1, i), δ_2(q_2, τ_1(q_1, i)))
let F = {(q_1, q_2) | q_2 ∈ F_2}
let M' = (Q, q_0, δ, F)
if [[M']] = D^ω then
    return "yes"
else
    find x(y)^ω ∈ (D^ω \ [[M']])
    let u(v)^ω = t(x(y)^ω)
    return "no" and (u, v)
```

types about L and eventually halts and outputs a DBW acceptor M such that $[\![M]\!] = L$. The algorithm runs in polynomial time iff there is a fixed polynomial p such that for every $L \in \mathbb{C}$, at every point the number of steps used by $\mathbf{A}$ is bounded by $p(n, m)$, where n is the size of the smallest DBW acceptor recognizing L, and m is the maximum length of any counterexample $\mathbf{A}$ has received up to that point.

The case of a learning algorithm for ω-tree languages is analogous, except that the examples and counterexamples are given by regular ω-tree automata, and the hypotheses provided to equivalence or subset queries are represented by DBT acceptors. We also consider cases in which the inputs to equivalence or subset queries may be NBW or NBT acceptors.

5 Framework of a reduction

Suppose $\mathbf{A}$ is a learning algorithm that uses membership and equivalence queries and exactly learns a class $\mathbb{C} \subseteq \mathbb{DBW}$. We shall describe an algorithm $\mathbf{A}_{Trees}$ that uses membership and equivalence queries and exactly learns the derived class $Trees_d(\mathbb{C})$ of ω-tree languages. Note that $Trees_d(\mathbb{C}) \subseteq \mathbb{DBT}$.

The algorithm $\mathbf{A}_{Trees}$ with target concept $Trees_d(L)$ simulates algorithm $\mathbf{A}$ with target concept L. In order to do so, $\mathbf{A}_{Trees}$ must correctly answer membership and equivalence queries from $\mathbf{A}$ about L by making one or more membership and/or equivalence queries of its own about $Trees_d(L)$. Before describing $\mathbf{A}_{Trees}$ we establish some basic results about regular ω-trees.

5.1 Testing acceptance of a regular ω-tree

We describe a polynomial time algorithm *Accepted?*(A_t, M) that takes as input a regular ω-tree t represented by a regular ω-tree automaton $A_t = (Q_1, q_{0,1}, \delta_1, \tau_1)$ and a DBW acceptor $M = (Q_2, q_{0,2}, \delta_2, F_2)$ and determines whether or not $M^{T,d}$ accepts t. If not, it also outputs a pair (u, v) of finite words such that $u(v)^\omega \in (paths(t) \setminus [\![M]\!])$.

We may assume M is complete by adding (if necessary) a new non-accepting sink state and directing all undefined transitions to the new state. We construct a DBW acceptor M' over the alphabet $D = \{1, \ldots, d\}$ by combining A_t and M as follows. The states are $Q = Q_1 \times Q_2$, the initial state is $q_0 = (q_{0,1}, q_{0,2})$, the set of accepting states is $F = \{(q_1, q_2) \mid q_2 \in F_2\}$, and the transition function δ is defined by $\delta((q_1, q_2), i) = (\delta_1(q_1, i), \delta_2(q_2, \tau_1(q_1, i)))$ for all $(q_1, q_2) \in Q$ and $i \in D$. For each transition, the output of the regular ω-tree automaton A_t is the input of the DBW acceptor M.

An infinite path π in t corresponds to an ω-word $z \in D^\omega$, giving the sequence of directions from the root. The unique run of M' on z traverses a sequence of states; if we project out the first component, we get the run of A_t on z, and if we project out the second component, we get the run of M on $t(\pi)$. Then $M^{T,d}$ accepts t iff M accepts $t(\pi)$ for every infinite path π, which is true iff $[\![M']\!] = D^\omega$. This in turn is true iff every nonempty accessible recurrent set of states in M' contains at least one element of F.

A set S of states is *recurrent* iff for all $q, q' \in S$, there is a nonempty finite word v such that $\delta(q, v) = q'$ and for every prefix u of v we have $\delta(q, u) \in S$. A set S of states is *accessible* iff for every $q \in S$ there exists a finite word u such that $\delta(q_0, u) = q$.

The algorithm to test whether $M^{T,d}$ accepts t first removes from the transition graph of M' all states that are not accessible. It then removes all states in F and tests whether there is any cycle in the remaining graph. If not, then $M^{T,d}$ accepts t. Otherwise, there is a state q in Q and finite words $x \in D^*$ and $y \in D^+$ such that $\delta(q_0, x) = q$ and $\delta(q, y) = q$ and none of the states traversed from q to q along the path y are in F. Thus, $x(y)^\omega$ is an ultimately periodic path π that does not visit F infinitely often, and letting $u(v)^\omega$ be $t(x(y)^\omega)$, we have $u(v)^\omega \in (\mathit{paths}(t) \setminus [\![M]\!])$, so the pair (u, v) is returned in this case. The required graph operations are standard and can be accomplished in time polynomial in $|M|$ and $|A_t|$.

5.2 Representing a language as paths of a tree

When the algorithm $\mathbf{A}_{\mathit{Trees}}$ makes a membership query about $\mathit{Trees}_d(L)$ with a regular ω-tree t, the answer is "yes" if $\mathit{paths}(t) \subseteq L$ and "no" otherwise. Thus, this query has the effect of a restricted subset query about L with $\mathit{paths}(t)$. But this does not give us restricted subset queries about L for arbitrary $\mathbb{DBW}$ languages. We show below that if $[\![M]\!]$ is a safety language, then it may be represented as $\mathit{paths}(t)$ for a regular ω-tree t.

An ω-word language L is a *safety language* iff L is a regular ω-word language and for every ω-word w not in L, there exists a finite prefix x of w such that no ω-word with prefix x is in L. A language is safety iff it is in the class $\mathbb{D}_\mathbb{W}\mathbb{CW}$. An alternative characterization is that there is an NBW acceptor $M = (Q, q_0, \delta, Q)$, all of whose states are accepting, such that $[\![M]\!] = L$. In this case, the acceptor is typically not complete (otherwise it recognizes Σ^ω). An example of a language in $\mathbb{D}_\mathbb{W}\mathbb{PW}$ that is not a safety language is $a^*b^*(a)^\omega$. Although b^ω is not in the language, every finite prefix b^k is a prefix of some ω-word in the language.

The following lemmas show a close connection between NBW acceptors of safety languages and $\mathit{paths}(t)$ for regular ω-trees t. The constructions in Lemmas 4 and 5 can be done in polynomial time. Lemma 5 is used in the proof of Lemma 9.

▶ **Lemma 4.** *If A_t is a regular ω-tree automaton representing an ω-tree t, then $\mathit{paths}(t)$ is a safety language recognizable by an NBW acceptor M with $|M| = |A_t|$.*

For the converse, representing a safety language as the paths of a regular ω-tree, we require a lower bound on d, the arity of the tree. If $M = (Q, q_0, \delta, F)$ is an NBW acceptor and $q \in Q$, we define the set of transitions out of q to be $\mathit{transitions}(q) = \{(\sigma, r) \mid \sigma \in \Sigma \wedge r \in \delta(q, \sigma)\}$. We define the *out-degree* of M to be the maximum over $q \in Q$ of the cardinality of $\mathit{transitions}(q)$.

▶ **Lemma 5.** *Let L be a safety language recognized by NBW acceptor $M = (Q, q_0, \delta, Q)$. Suppose the out-degree of M is at most d. Then there is a d-ary regular ω-tree t such that $\mathit{paths}(t) = L$, and t is representable by A_t with $|A_t| = |M|$.*

The NBW acceptor in the proof of Lemma 4 can be determinized via the subset construction to give a DBW acceptor of size at most $2^{|A_t|}$ recognizing the same language. In

Algorithm 2 : $\mathbf{A}_{Trees}$

Require: Learning algorithm $\mathbf{A}$ for $\mathbb{C}$;
MQ and EQ access to $Trees_d(L)$ for $L \in \mathbb{C}$
Ensure: Acceptor $M^{T,d}$ such that $[\![M^{T,d}]\!] = Trees_d(L)$

while $\mathbf{A}$ has not halted **do**
  if next step of $\mathbf{A}$ is not a query **then**
    simulate next step of $\mathbf{A}$
  else if $\mathbf{A}$ asks MQ(u, v) about L **then**
    answer $\mathbf{A}$ with MQ$(tree_d(u(v)^\omega))$ about $Trees_d(L)$
  else if $\mathbf{A}$ asks EQ(M) about L **then**
    ask EQ$(M^{T,d})$ about $Trees_d(L)$
    if EQ$(M^{T,d})$ answer is "yes" **then**
      return $M^{T,d}$ and halt
    else {EQ$(M^{T,d})$ answer is counterexample tree t given by A_t}
      if $Accepted?(A_t, M)$ returns "no" with value (u, v) **then**
        answer $\mathbf{A}$ with (u, v)
      else {$Accepted?(A_t, M)$ returns "yes"}
        let $M' = acceptor(A_t)$
        for all accepting states q of M' **do**
          simulate in parallel $Findctrex(M', q)$
          terminate all computations and answer $\mathbf{A}$ with the first (u, v) returned
[$\mathbf{A}$ halts with output M]
return $M^{T,d}$ and halt

the worst case this exponential blow up in converting a regular ω-tree automaton to a DBW acceptor is necessary, as shown by the following.

▶ **Lemma 6.** *There exists a family of regular ω-trees $t_1, t_2, \ldots$ such that t_n can be represented by a regular ω-tree automaton of size $n + 2$, but the smallest DBW acceptor recognizing $paths(t_n)$ has size at least 2^n.*

If t is a d-ary regular ω-tree represented by the regular ω-tree automaton A_t, then $acceptor(A_t)$ denotes the NBW acceptor M recognizing $paths(t)$ constructed from A_t in the proof of Lemma 4. Note that the out-degree of $acceptor(A_t)$ is at most d.

If M is an NBW acceptor such that $[\![M]\!]$ is a safety language and the out-degree of M is at most d, then $tree_d(M)$ denotes the regular ω-tree automaton A_t constructed from M in the proof of Lemma 5. We also use the notation $tree_d(L)$ if L is a safety language and the implied acceptor for L is clear.

For example, given finite words $u \in \Sigma^*$ and $v \in \Sigma^+$, the singleton set containing $u(v)^\omega$ is a safety language recognized by a DBW of out-degree 1 and size linear in $|u| + |v|$. Then $tree_d(u(v)^\omega)$ represents the d-ary tree all of whose infinite paths are labeled with $u(v)^\omega$.

6 The algorithm $\mathbf{A}_{Trees}$

We now describe the algorithm $\mathbf{A}_{Trees}$, which learns $Trees_d(L)$ by simulating the algorithm $\mathbf{A}$ and answering the membership and equivalence queries of $\mathbf{A}$ about L. It is summarized in Algorithm 2 (at the end of the document).

If $\mathbf{A}$ asks a membership query with (u, v) then $\mathbf{A}_{Trees}$ constructs the regular ω-tree automaton $tree_d(u(v)^\omega)$ representing the d-ary regular ω-tree all of whose infinite paths are labeled $u(v)^\omega$, and makes a membership query with $tree_d(u(v)^\omega)$. Because $u(v)^\omega \in L$ iff the tree represented by $tree_d(u(v)^\omega)$ is in $Trees_d(L)$, the answer to the query about $tree_d(u(v)^\omega)$ is simply given to $\mathbf{A}$ as the answer to its membership query about (u, v).

For an equivalence query from $\mathbf{A}$ specified by a DBW acceptor M, the algorithm $\mathbf{A}_{Trees}$ constructs the corresponding DBT acceptor $M^{T,d}$, which recognizes $Trees_d([\![M]\!])$, and makes an equivalence query with $M^{T,d}$. If the answer is "yes", the algorithm $\mathbf{A}_{Trees}$ has succeeded in learning the target ω-tree language $Trees_d(L)$ and outputs $M^{T,d}$ and halts. Otherwise,

the counterexample returned is a regular ω-tree t in $[\![M^{T,d}]\!] \oplus \mathit{Trees}_d(L)$, represented by a regular ω-tree automaton A_t. A call to the algorithm $\mathit{Accepted?}(A_t, M)$ determines whether $M^{T,d}$ accepts t. If $M^{T,d}$ rejects t, then $t \in \mathit{Trees}_d(L)$ and t is a *positive counterexample*. If $M^{T,d}$ accepts t, then $t \notin \mathit{Trees}_d(L)$ and t is a *negative counterexample*. We consider these two cases.

If t is a positive counterexample then we know that $t \in \mathit{Trees}_d(L)$ and therefore $\mathit{paths}(t) \subseteq L$. Because $t \notin [\![M^{T,d}]\!]$, the acceptor M must reject at least one infinite path in t. In this case, the algorithm $\mathit{Accepted?}(A_t, M)$ returns a pair of finite words (u, v) such that $u(v)^\omega \in (\mathit{paths}(t) \setminus [\![M]\!])$, and therefore $u(v)^\omega \in (L \setminus [\![M]\!])$. The algorithm $\mathbf{A}_{\mathit{Trees}}$ returns the positive counterexample (u, v) to $\mathbf{A}$ in response to its equivalence query with M.

If t is a negative counterexample, that is, $t \in ([\![M^{T,d}]\!] \setminus \mathit{Trees}_d(L))$, then we know that $\mathit{paths}(t) \subseteq [\![M]\!]$, but at least one element of $\mathit{paths}(t)$ is not in L, so $([\![M]\!] \setminus L) \neq \emptyset$. Ideally, we would like to extract an ultimately periodic ω-word $u(v)^\omega \in (\mathit{paths}(t) \setminus L)$ and provide (u, v) to $\mathbf{A}$ as a negative counterexample in response to its equivalence query with M.

If we could make an *unrestricted subset query* with $\mathit{paths}(t)$ about L, then the counterexample returned would be precisely what we need.

As noted previously, if t is any regular ω-tree then we can simulate a restricted subset query with $\mathit{paths}(t)$ about L by making a membership query with t about $\mathit{Trees}_d(L)$, because $\mathit{paths}(t) \subseteq L$ iff $t \in \mathit{Trees}_d(L)$. In order to make use of this, we next show how to use restricted subset queries about L to implement an unrestricted subset query about L.

6.1 Restricted subset queries

To establish basic techniques, we show how to reduce unrestricted subset queries to restricted subset queries for nondeterministic or deterministic finite acceptors over finite words. Suppose $L \subseteq \Sigma^*$ and we may ask restricted subset queries about L. In such a query, the input is a nondeterministic (resp., deterministic) finite acceptor M, and the answer is "yes" if $[\![M]\!]$ is a subset of L, and "no" otherwise. If the answer is "no", we show how to find a shortest counterexample $u \in ([\![M]\!] \setminus L)$ in time polynomial in $|M|$ and $|u|$.

▶ **Theorem 7.** *There is an algorithm $\mathbf{R}^*$ with input M, an NFW (resp., DFW), and restricted subset query access to a language L with NFW (resp., DFW) acceptors as inputs, that correctly answers the unrestricted subset query with M about L. If L is recognized by a DFW T_L, then $\mathbf{R}^*(M)$ runs in time bounded by a polynomial in $|M|$ and $|T_L|$.*[1]

The idea of the proof is to first establish the minimal length ℓ of a counterexample, and then try to extend the prefix ϵ letter by letter until obtaining a full length minimal counterexample (see Appendix A). Note that trying to establish a prefix of a counterexample letter by letter, without obtaining a bound first, may not terminate. For instance, if $L = \Sigma^* \setminus a^*b$, one can establish the sequence of prefixes $\epsilon, a, aa, aaa, \ldots$ and never reach a counterexample.

We now turn to the ω-word case.

▶ **Theorem 8.** *There is an algorithm $\mathbf{R}^\omega$ with input M and restricted subset query access about L, (a language recognized by a DBW acceptor T_L) that correctly answers the unrestricted subset query with M about L. $\mathbf{R}^\omega(M)$ runs in time bounded by a polynomial in $|M|$ and $|T_L|$. If M is a DBW acceptor, then all the restricted subset queries will also be with DBW acceptors.*

[1] The cardinality of Σ is treated as a constant.

For the sake of generality, the proof considers subset queries with NBW acceptors. The procedure $\mathbf{R}^{\omega}(M)$ takes as input an NBW acceptor M, and has restricted subset query access (with NBW acceptors as inputs) to L; it is summarized in Algorithm 3. It first asks a restricted subset query with M about L, returning the answer "yes" if its query is answered "yes". Otherwise, for each $q \in F$, it constructs the acceptor $M_q = (Q, q_0, \delta, \{q\})$ with the single accepting state q and asks a restricted subset query with M_q about L, until the first query answered "no". There will be at least one such query answered "no" because any element of $(\llbracket M \rrbracket \setminus L)$ must visit at least one accepting state q of M infinitely many times, and will therefore be in $\llbracket M_q \rrbracket$. The procedure $\mathbf{R}^{\omega}(M)$ then calls the procedure *Findctrex*(M, q) to find a counterexample to return – i.e. a pair (u, v) such that $u(v)^{\omega} \in (\llbracket M \rrbracket \setminus L)$.

6.2 Producing a counterexample

The first challenge encountered in producing a counterexample, in comparison to the finite word case, is that one needs to work out both the period and the prefix of the counterexample to be found, and the two are correlated. Define $L_{q_0,q}$ to be the set of finite words that lead from the initial state q_0 to the state q in M, and define $L_{q,q}$ to be the set of nonempty finite words that lead from q back to q in M. Because the language $L_{q_0,q} \cdot (L_{q,q})^{\omega}$ is exactly the set of strings recognized by M_q, we know that $L_{q_0,q} \cdot (L_{q,q})^{\omega} \setminus L \neq \emptyset$.

The procedure *Findctrex*(M, q) first finds a suitable period, corresponding to a bounded size of a prefix yet to be found, and then finds a prefix of that size in a similar manner to the finite word case.

Since finding the period is more challenging than the prefix, we explain the procedure *Findprefix*(M, q, v) first. The procedure *Findprefix*(M, q, v), summarized in Algorithm 5, finds a prefix word u given a period word v which loops on state q and is guaranteed to be a period of a valid counterexample. It first finds a length k such that there exists $u \in L_{q_0,q}$ of length k such that $uv^{\omega} \notin L$. Then it finds such a word u symbol by symbol. Note that it uses length and prefix restricted versions of $L_{q_0,q}$.

Finding the periodic part is much more challenging. Indeed, even if one knows that there is a period of the form $(a\Sigma^{\ell})^{\omega}$ for some ℓ then the size of the smallest period may be bigger than $\ell + 1$. For instance, if $L = \Sigma^{\omega} \setminus (abbaccadd)^{\omega}$ then there is a period of the form $(a\Sigma^2)^{\omega}$ but the shortest period of a counterexample is of size 9.

Procedure *Findperiod*(M, q), summarized in Algorithm 6, starts from the condition

$$L_{q_0,q} \cdot (L_{q,q})^{\omega} \setminus L \neq \emptyset$$

and finds a sequence of words $v_1, v_2, \ldots \in L_{q,q}$ such that for each $n \geq 1$,

$$L_{q_0,q} \cdot (v_1 v_2 \cdots v_n \cdot L_{q,q})^{\omega} \setminus L \neq \emptyset.$$

For a sufficiently long such sequence, there exists a subsequence $v = (v_i \cdots v_j)$ that is a suitable period word, as we illustrate in Appendix C.2.

The procedure *Nextword*(M, q, y), summarized in Algorithm 7, is called with $y = v_1 v_2 \cdots v_n$ and finds a suitable next word v_{n+1}. After determining a length ℓ, it repeatedly calls the procedure *Nextsymbol*(M, q, y, ℓ, v') to determine the next symbol of a suitable word of length ℓ.

The procedure *Nextsymbol*(M, q, y, ℓ, v'), summarized in Algorithm 8, is called to find a feasible next symbol with which to extend v' in the procedure *Nextword*.

7 Correctness

The main hurdle in proving correctness of algorithm $\mathbf{A}_{Trees}$ is to prove Theorem 8. The polynomial bound in the proof of Theorem 8 is obtained through a sequence of lemmas (Lemmas 12 to 14) bounding the size of the acceptors used in $\mathbf{A}_{Trees}$ subprocedures and the length restrictions and running time in calls to RSQ made by these procedures. Appendix C.1 deals with bounding the acceptors, and Appendix C.2 deals with the more challenging part, providing the length restrictions. In the following we conclude with the theorem stating the correctness of algorithm $\mathbf{A}_{Trees}$.

The lemmas established in the previous subsection also show the correctness and running time of $Findctrex(M', q)$ when called by $\mathbf{A}_{Trees}$, provided that each RSQ about L is correctly answered and q satisfies the precondition of *Findctrex*.

To complete the consideration of representation issues, we must prove that $\mathbf{A}_{Trees}$ can successfully simulate *Findctrex* as stated in Lemma 9.

▶ **Lemma 9.** *When $\mathbf{A}_{Trees}$ simulates $Findctrex(M', q)$ in response to a negative counterexample t, every* RSQ *can be simulated with a* MQ *about $Trees_d(L)$.*

If q does not satisfy the precondition of *Findctrex*, then the procedure may run forever. However, at least one accepting state q satisfies the precondition, so at least one simulation will halt and return (u, v), at which point $\mathbf{A}_{Trees}$ terminates the other simulations. This concludes the proof of the reduction given by $\mathbf{A}_{Trees}$, whose general statement is given in Theorem 10.

▶ **Theorem 10.** *Suppose $\mathbb{C} \subseteq \mathbb{DBW}$ and $\mathbf{A}$ is a polynomial time algorithm that learns class $\mathbb{C}$ using membership and equivalence queries. Then for every positive integer d there is a polynomial time algorithm $\mathbf{A}_{Trees}$ that learns $Trees_d(\mathbb{C})$ using membership and equivalence queries.*

This theorem, together with Maler and Pnueli's [15] polynomial time algorithm to learn the class of weak regular ω-word languages using membership and equivalence queries proves our main result – Theorem 3.

8 Discussion

We have shown that if $\mathbb{C} \subseteq \mathbb{DBW}$ can be learned in polynomial time with membership and equivalence queries, then $Trees_d(\mathbb{C})$ can be learned in polynomial time with membership and equivalence queries for all $d \geq 1$. Consequently, there is a polynomial time algorithm to learn $Trees_d(\mathbb{DwPW})$ with membership and equivalence queries. We have also shown that there are polynomial time algorithms that implement unrestricted subset queries using restricted subset queries for $\mathbb{DFW}$, $\mathbb{NFW}$, $\mathbb{DBW}$ and $\mathbb{NBW}$.

One open question is whether there is an interesting subclass of $\mathbb{DBW}$ that is larger than $\mathbb{DwPW}$ but still learnable in polynomial time using membership and equivalence queries, to which Theorem 10 would also apply.

References

1 D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.
2 D. Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1988.
3 D. Angluin. Negative results for equivalence queries. *Mach. Learn.*, 5(2):121–150, 1990.

4 D. Angluin. The class DBW ∩ DCW of ω-languages is identifiable in the limit from positive data and membership queries with polynomial time and data. Technical Report YALEU/DCS/TR-1528, Dept. of Computer Science, Yale University, August 2016.

5 D. Angluin and D. Fisman. Learning regular omega languages. *Theor. Comput. Sci.*, 650:57–72, 2016.

6 U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Automata, Languages, and Programming, 40th Inter. Colloq., ICALP*, pages 89–100, 2013.

7 H. Calbrix, M. Nivat, and A. Podelski. Ultimately periodic words of rational ω-languages. In *Proc. of the 9th Inter. Conf. on Mathematical Foundations of Programming Semantics*, pages 554–566, 1994.

8 J. M. Cobleigh, D. Giannakopoulou, and C. S. Păsăreanu. Learning assumptions for compositional verification. In *Proc. of the 9th Inter. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, pages 331–346, 2003.

9 A. Farzan, Y. Chenand, E.M. Clarke, Y. Tsay, and B. Wang. Extending automated compositional verification to the full class of omega-regular languages. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 4963 of *LNCS*, pages 2–17, 2008.

10 E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Springer, 2002.

11 M. Jayasrirani, M. Humrosia Begam, and D. Gnanaraj Thomas. Learning of regular omega-tree languages. In *Grammatical Inference: Algorithms and Applications, 9th Inter. Colloq., ICGI*, pages 295–297, 2008.

12 D. Kuperberg and M. Skrzypczak. On determinisation of good-for-games automata. In *Automata, Languages, and Programming, 42nd Inter. Colloq., ICALP*, pages 299–310, 2015.

13 O. Kupferman, S. Safra, and M. Y. Vardi. Relating word and tree automata. *Annals of Pure and Applied Logic*, 138:126–146, 2006.

14 C. Löding. Automata on infinite trees. Available online from author, 2011.

15 O. Maler and A. Pnueli. On the learnability of infinitary regular sets. *Inform. Comput.*, 118(2):316–326, 1995.

16 Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. of the Ninth Annual ACM Symposium on Principles of Distributed Computing, PODC*, pages 377–410, 1990.

17 D. Niwiński and I. Walukiewicz. Relating hierarchies of word and tree automata. In *In Symposium on Theoretical Aspects in Computer Science, STACS, LNCS 1373*, pages 320–331. Springer-Verlag, 1998.

18 D. Perrin and J.-É. Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.

19 M. Rabin. *Automata on infinite objects and Church's problem*. Regional Conf. series in mathematics. Conf. Board of the Math. Sci. by the American Mathematical Society, 1972.

20 F. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017.

21 K. W. Wagner. A hierarchy of regular sequence sets. In *4th Symposium on Mathematical Foundations of Computer (MFCS)*, pages 445–449, 1975.

A Proof of Theorem 7

To prove Theorem 7 we first construct an acceptor $M_{\ell,v}$ for $[\![M]\!][\ell, v]$, the length and prefix restricted version of $[\![M]\!]$, given M, ℓ and v as inputs.

▶ **Lemma 11.** *There is a polynomial time algorithm to construct an acceptor $M_{\ell,v}$ for $[\![M]\!][\ell, v]$ given a NFW acceptor M, a nonnegative integer ℓ and a finite word v, such that*

1. *$M_{\ell,v}$ has at most one accepting state, which has no out-transitions,*
2. *the out-degree of $M_{\ell,v}$ is at most the out-degree of M,*
3. *$M_{\ell,v}$ is deterministic if M is deterministic.*

Proof of Theorem 7. For input M, define $M_{[\ell,v]}$ to be the finite acceptor constructed by the algorithm of Lemma 11 to recognize the length and prefix restricted language $[\![M]\!][\ell, v]$.

For $\ell = 0, 1, 2, \ldots$, ask a restricted subset query with $M_{[\ell,\varepsilon]}$, until the first query answered "no". At this point, ℓ is the shortest length of a counterexample in $([\![M]\!] \setminus L)$. Then a counterexample u of length ℓ is constructed symbol by symbol.

Assume we have found a prefix u' of a counterexample of length ℓ in $([\![M]\!] \setminus L)$, with $|u'| < \ell$. For each symbol $\sigma \in \Sigma$ we ask a restricted subset query with $M_{[\ell,u'\sigma]}$, until the first query answered "no". At this point, u' is extended to $u'\sigma$. If the length of $u'\sigma$ is now ℓ, then $u = u'\sigma$ is the desired counterexample; otherwise, we continue extending u'.

Note that if the input M is deterministic, then all of the restricted subset queries are made with deterministic finite acceptors. If L is recognized by a deterministic finite acceptor T_L, then the value of ℓ is bounded by $|M| \cdot |T_L|$, and the algorithm runs in time bounded by a polynomial in $|M|$ and $|T_L|$. ◀

B Algorithms

Algorithm 3 : $\mathbf{R}^{\omega}(M)$, implementing $\textsc{usq}(M)$

Require: RSQ access to L;
$M = (Q, q_0, \delta, F)$, an NBW acceptor
Ensure: "yes" if $[\![M]\!] \subseteq L$, else "no" and (u, v) s.t. $u(v)^{\omega} \in ([\![M]\!] \setminus L)$

if $\textsc{rsq}(M)$ = "yes" **then**
  return "yes"
else
  find $q \in F$ such that $\textsc{rsq}(M_q)$ = "no"
  return "no" and *Findctrex*(M, q)

Algorithm 4 : *Findctrex*(M, q)

Require: RSQ access to L;
$M = (Q, q_0, \delta, F)$, an NBW acceptor;
$q \in F$;
$L_{q_0,q} \cdot (L_{q,q})^{\omega} \setminus L \neq \emptyset$
Ensure: (u, v) such that $u(v)^{\omega} \in (L_{q_0,q} \cdot (L_{q,q})^{\omega} \setminus L)$

let v = *Findperiod*(M, q)
let u = *Findprefix*(M, q, v)
return (u, v)

Algorithm 5 : *Findprefix*(M, q, v)

Require: RSQ access to L;
$M = (Q, q_0, \delta, F)$, an NBW acceptor;
$q \in F$;
$v \in L_{q,q}$;
$L_{q_0,q} \cdot (v)^\omega \setminus L \neq \emptyset$
Ensure: $u \in L_{q_0,q}$ such that $u(v)^\omega \in (L_{q_0,q} \cdot (v)^\omega \setminus L)$

search for nonnegative integer k such that $\text{RSQ}(L_{q_0,q}[k] \cdot (v)^\omega) =$ "no"
let $u = \varepsilon$
while $|u| < k$ **do**
  find $\sigma \in \Sigma$ such that $\text{RSQ}(L_{q_0,q}[k, u\sigma] \cdot (v)^\omega) =$ "no"
  set $u = u \cdot \sigma$
return u

Algorithm 6 : *Findperiod*(M, q)

Require: RSQ access to L;
$M = (Q, q_0, \delta, F)$, an NBW acceptor;
$q \in F$;
$L_{q_0,q} \cdot (L_{q,q})^\omega \setminus L \neq \emptyset$
Ensure: $v \in L_{q,q}$ such that $L_{q_0,q} \cdot (v)^\omega \setminus L \neq \emptyset$

let $y = \varepsilon$
for all integers $n = 1, 2, 3, \ldots$ **do**
  let $v_n = \mathit{Nextword}(M, q, y)$
  set $y = y \cdot v_n$
  for integers i, j with $1 \leq i \leq j \leq n$ **do**
    for $k = 0$ to $n|M|$ **do**
      if $\text{RSQ}(L_{q_0,q}[k] \cdot (v_i \cdots v_j)^\omega) =$ "no" **then**
        return $v = v_i \cdots v_j$

Algorithm 7 : *Nextword*(M, q, y)

Require: RSQ access to L;
$M = (Q, q_0, \delta, F)$, an NBW acceptor;
$q \in F$;
$y \in L_{q,q}$ or $y = \varepsilon$;
$L_{q_0,q} \cdot (y \cdot L_{q,q})^\omega \setminus L \neq \emptyset$
Ensure: $v' \in L_{q,q}$ such that $L_{q_0,q} \cdot (y \cdot v' \cdot L_{q,q})^\omega \setminus L \neq \emptyset$

search for integers $k, \ell \geq 0$ s.t. $\text{RSQ}(L_{q_0,q}[k] \cdot (y \cdot L_{q,q}[\ell])^\omega) =$ "no"
let $v' = \varepsilon$
while $|v'| < \ell$ **do**
  let $\sigma = \mathit{Nextsymbol}(M, q, y, \ell, v')$
  set $v' = v' \cdot \sigma$
return v'

Algorithm 8 : *Nextsymbol*(M, q, y, ℓ, v')

Require: RSQ access to L;
$M = (Q, q_0, \delta, F)$, an NBW acceptor;
$q \in F$;
$y \in L_{q,q}$;
$v' \in \Sigma^*$, $|v'| < \ell$;
$L_{q_0,q} \cdot (y \cdot L_{q,q}[\ell, v'] \cdot L_{q,q})^\omega \setminus L \neq \emptyset$
Ensure: $\sigma \in \Sigma$ such that $L_{q_0,q} \cdot (y \cdot L_{q,q}[\ell, v'\sigma] \cdot L_{q,q})^\omega \setminus L \neq \emptyset$

find integers $k \geq 0$, $m \geq 1$, and $\sigma \in \Sigma$ such that
$\text{RSQ}(L_{q_0,q}[k] \cdot (y \cdot L_{q,q}[\ell, v'\sigma] \cdot L_{q,q}[m])^\omega) =$ "no"
return σ

C Correctness and bounds

C.1 Bounding the acceptors

We turn to the representation (as NBW or DBW acceptors) of the languages used in restricted subset queries by $\mathbf{R}^\omega(M)$ and its subprocedures. We consider the size, out-degree, and time to construct the acceptors.

In $\mathbf{R}^\omega(M)$, there is a restricted subset query with M itself, and if that query is answered "no", a sequence of restricted subset queries with M_q for accepting states q until an answer of "no". Clearly, if M is an NBW acceptor, each M_q is an NBW acceptor of the same size and out-degree and is easily constructed from M, and similarly if M is a DBW acceptor.

The restricted subset queries made in *Findctrex* and its subprocedures are of the form $P \cdot (S)^\omega$, where P is a length and prefix restricted version of $L_{q_0,q}$ and S is a concatenation of (at most) a finite word and two length and prefix restricted versions of $L_{q,q}$. Thus we consider the operations of concatenation and ω-repetition of regular languages of finite words.

These operations are particularly simple for DFW or NFW acceptors in *special form*, that is, containing at most one accepting state, which has no out-transitions defined. In general, any NFW acceptor can be converted to special form, possibly at the cost of increasing its out-degree. A regular language of finite words is recognized by a DFW acceptor in special form iff it is prefix-free.

However, if M is an NBW (resp., DBW) acceptor, then the finite word languages $L_{q_0,q}$ and $L_{q,q}$ are recognized by easily constructed NFW (resp., DFW) acceptors of size at most $|M|$ and out-degree at most the out-degree of M. Lemma 11 shows that the length and prefix restricted versions of $L_{q_0,q}$ and $L_{q,q}$ are recognized by NFW (resp., DFW) acceptors in special form which may be constructed in time polynomial in $|M|$, ℓ, and $|v|$ and have out-degree at most the out-degree of M.

▶ **Lemma 12.** *Suppose M_1 is an NFW acceptor in special form and M_2 is an NFW or NBW acceptor. Then an acceptor M for $[\![M_1]\!] \cdot [\![M_2]\!]$ can be constructed such that*

1. $|M| \leq |M_1| + |M_2|$,
2. *the out-degree of M is at most the maximum of out-degrees of M_1 and M_2,*
3. *M can be constructed in polynomial time,*
4. *M is deterministic if M_1 and M_2 are deterministic,*
5. *M is an NFW in special form if M_2 is an NFW in special form.*

▶ **Lemma 13.** *Suppose M_1 is an NFW acceptor in special form. Then an NBW acceptor M for $[\![M_1]\!]^\omega$ can be constructed such that*

1. $|M| \leq |M_1|$,
2. *the out-degree of M is at most the out-degree of M_1,*
3. *M can be constructed in polynomial time,*
4. *M is deterministic if M_1 is deterministic.*

The above give us the following corollary for the procedure $\mathbf{R}^\omega$.

▶ **Corollary 14.** *When the input to $\mathbf{R}^\omega(M)$ is an NBW (resp., DBW) acceptor M, each* RSQ *can be made with an NBW (resp., DBW) acceptor whose out-degree is at most the out-degree of M and can be constructed in time polynomial in $|M|$ and parameters giving the length restrictions and the lengths of any words that appear.*

C.2 Length restrictions and time bounds

We now turn to establish the correctness and running time of the subprocedures. The first two lemmas allow us to bound the parameters giving the length restrictions in inputs to RSQ.

▶ **Lemma 15.** *Let $S \subseteq L_{q,q}$ and suppose $L_{q_0,q} \cdot (S)^\omega \setminus L \neq \emptyset$. Then for some $k < |M| \cdot |T_L|$ we have $L_{q_0,q}[k] \cdot (S)^\omega \setminus L \neq \emptyset$.*

▶ **Lemma 16.** *Let $S \subseteq L_{q,q}$ and suppose $L_{q_0,q} \cdot (S \cdot L_{q,q})^\omega \setminus L \neq \emptyset$. Then for some $k, \ell < |M| \cdot |T_L|$, we have that $L_{q_0,q}[k] \cdot (S \cdot L_{q,q}[\ell])^\omega \setminus L \neq \emptyset$.*

We now prove the correctness and polynomial running time of *Findprefix* and *Findperiod*, which establishes the correctness and polynomial running time of *Findctrex*.

▶ **Lemma 17.** *Assume $v \in L_{q,q}$ is such that*

$$L_{q_0,q} \cdot (v)^\omega \setminus L \neq \emptyset.$$

Then in time polynomial in $|M|$, $|T_L|$ and $|v|$, the procedure Findprefix(M, q, v) returns a word $u \in L_{q_0,q}$ such that

$$u(v)^\omega \in (L_{q_0,q} \cdot (v)^\omega \setminus L).$$

The procedure *Findperiod* depends on the procedures *Nextword* and *Nextsymbol*. The next lemma establishes the correctness and running time of the procedure *Nextsymbol*.

▶ **Lemma 18.** *Suppose ℓ is a positive integer, $y \in L_{q,q}$ or $y = \varepsilon$ and $v' \in \Sigma^*$ is such that $|v'| < \ell$ and we have*

$$L_{q_0,q} \cdot (y \cdot L_{q,q}[\ell, v'] \cdot L_{q,q})^\omega \setminus L \neq \emptyset.$$

Then in time polynomial in $|M|$, $|T_L|$, $|y|$ and ℓ, Nextsymbol(M, q, y, ℓ, v') finds a symbol $\sigma \in \Sigma$ such that

$$L_{q_0,q} \cdot (y \cdot L_{q,q}[\ell, v'\sigma] \cdot L_{q,q})^\omega \setminus L \neq \emptyset.$$

▶ **Lemma 19.** *Suppose $y \in L_{q,q}$ or $y = \varepsilon$ is such that*

$$L_{q_0,q} \cdot (y \cdot L_{q,q})^\omega \setminus L \neq \emptyset.$$

Then in time bounded by a polynomial in $|M|$, $|T_L|$ and $|y|$, Nextword(M, q, y) returns a word $v' \in L_{q,q}$ of length bounded by $|M| \cdot |T_L|$ such that

$$L_{q_0,q} \cdot (yv' \cdot L_{q,q})^\omega \setminus L \neq \emptyset.$$

The next lemma shows that *Findperiod* calls *Nextword* at most $|T_L|$ times.

▶ **Lemma 20.** *Suppose $v_1, v_2, \ldots, v_n \in L_{q,q}$ are such that*

$$L_{q_0,q} \cdot (v_1 v_2 \cdots v_n \cdot L_{q,q})^\omega \setminus L \neq \emptyset.$$

Also suppose that the number of states of T_L is less than n. Then there exist integers i and j with $1 \leq i \leq j \leq n$ such that

$$L_{q_0,q} \cdot (v_i v_{i+1} \cdots v_j)^\omega \setminus L \neq \emptyset.$$

The final lemma establishes the correctness and polynomial running time of the procedure *Findperiod*.

▶ **Lemma 21.** *Suppose* $L_{q_0,q} \cdot (L_{q,q})^\omega \setminus L \neq \emptyset$. *Then, in polynomial time in* $|M|$ *and* $|T_L|$, *the procedure* Findperiod(M, q) *with restricted query access to* L *returns a period word* v *satisfying the condition*

$$L_{q_0,q} \cdot (v)^\omega \setminus L \neq \emptyset.$$

These lemmas combine to prove Theorem 8, giving a polynomial time reduction of unrestricted subset queries to restricted subset queries for NBW acceptors (resp., DBW acceptors.)

Extending Two-Variable Logic on Trees*†

Bartosz Bednarczyk[1], Witold Charatonik[2], and Emanuel Kieroński[3]

1 **University of Wrocław, Wrocław, Poland**
bbednarczyk@stud.cs.uni.wroc.pl
2 **University of Wrocław, Wrocław, Poland**
wch@cs.uni.wroc.pl
3 **University of Wrocław, Wrocław, Poland**
kiero@cs.uni.wroc.pl

Abstract

The finite satisfiability problem for the two-variable fragment of first-order logic interpreted over trees was recently shown to be ExpSpace-complete. We consider two extensions of this logic. We show that adding either additional binary symbols or counting quantifiers to the logic does not affect the complexity of the finite satisfiability problem. However, combining the two extensions and adding both binary symbols and counting quantifiers leads to an explosion of this complexity. We also compare the expressive power of the two-variable fragment over trees with its extension with counting quantifiers. It turns out that the two logics are equally expressive, although counting quantifiers do add expressive power in the restricted case of unordered trees.

1998 ACM Subject Classification F.4 Mathematical Logic and Formal Languages

Keywords and phrases two-variable logic, trees, satisfiability, expressivity, counting quantifiers

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.11

1 Introduction

Two-variable logics. Two-variable logic, FO^2, is one of the most prominent decidable fragments of first-order logic. It is important in computer science because of its decidability and connections with other formalisms like modal, temporal and description logics or query languages. For example, it is known that FO^2 over words can express the same properties as unary temporal logic [10] and FO^2 over trees is precisely as expressive as the navigational core of XPath, a query language for XML documents [20]. The complexity of the satisfiability problem for FO^2 over words and trees, respectively, is studied in [10], and [2]. Namely, it is shown that its satisfiability problem over words is NExpTime-complete and over trees—ExpSpace-complete.

On the other hand, FO^2 cannot express that a structure is a word or a tree and it cannot express that a relation is transitive, an equivalence or an order. This led to extensive studies of FO^2 over various classes of structures, where some distinguished relational symbols are interpreted in a special way, e.g., as equivalences or linear orders. The finite satisfiability problem for FO^2 remains decidable over structures where one [17] or two relation symbols [18] are interpreted as equivalence relations; where one [21] or two relations are interpreted as linear orders [25, 27]; where two relations are interpreted as successors of two linear orders [19, 11, 8]; where one relation is interpreted as linear order, one as its successor

* Supported by the Polish National Science Centre grant No. 2016/21/B/ST6/01444.
† For missing proofs see [1].

and another one as equivalence [3]; where an equivalence closure can be applied to two binary predicates [16]; where deterministic transitive closure can be applied to one binary relation [6]. It is known that the finite satisfiability problem is undecidable for FO^2 with two transitive relations [14], with three equivalence relations [17], with one transitive and one equivalence relation [18], with three linear orders [15], with two linear orders and their two corresponding successors [19]. A summary of complexity results for extensions of FO^2 with binary predicates being the order relations can be found in [27].

In the context of extensions of FO^2 it is enough to consider relational signatures with symbols of arity at most two [12]. Some of the above mentioned decidability results, e.g., [2, 25, 19, 11, 3, 6], are obtained under the restriction that besides the distinguished binary symbols interpreted in a special way there are no other binary predicates in the signature; some, like [17, 18, 21, 8, 16, 27] are valid in the general setting. In undecidability results additional binary predicates are usually not necessary.

Another decidable extension of FO^2 is the two-variable fragment with *counting quantifiers*, C^2, where quantifiers of the form $\exists^{\leq k}$, $\exists^{=k}$, $\exists^{\geq k}$ are allowed. The finite satisfiability problem for C^2 was proved to be decidable and NExpTime-complete (both under unary and binary encoding of numbers in counting quantifiers) in [13, 22, 23]. There are also decidable extensions of C^2 with special interpretations of binary symbols: in [8] two relation symbols are interpreted as child relations in two forests (which subsumes the case of two successor relations on two linear orders), in [24] one symbol is interpreted as equivalence relation and in [7] one symbol is interpreted as linear order (and the case with two linear orders is undecidable).

Our contribution. In this paper we extend the main result from [2], namely ExpSpace-completeness of the satisfiability problem for FO^2 interpreted over finite trees without additional binary symbols. We consider two extensions of this logic. We show that adding either additional, uninterpreted binary symbols or counting quantifiers to the logic does not increase the complexity of the satisfiability problem. However, when we combine the two extensions and add both binary symbols and counting quantifiers then the complexity explodes and the problem is at least as hard as the emptiness problem for vector addition tree automata [9]. Since decidability of emptiness of vector addition tree automata is a long-standing open problem, showing decidability of C^2 over trees with additional binary symbols is unlikely in nearest future.

Let us recall that the situation is similar to the case of finite words: FO^2 with a linear order and the induced successor relation remains NExpTime-complete when extended either with additional binary relations [27] or with counting quantifiers [7]. Combining both additional ingredients gives a logic which is equivalent to the emptiness problem of multicounter automata [7], a problem which is known to be decidable, but for which no algorithm of elementary complexity is known.

We additionally compare the expressive power of the two-variable fragment over trees with its extension with counting quantifiers. It is not difficult to see that FO^2 over unordered trees cannot count and thus C^2 is strictly more expressive in this case. However, the presence of order in form of sibling relations gives FO^2 the ability of counting and makes the two logics equally expressive.

2 Preliminaries

2.1 Logics, trees and atomic types

We work with signatures of the form $\tau = \tau_0 \cup \tau_{nav} \cup \tau_{com}$, where τ_0 is a set of unary symbols, and $\tau_{nav} \subseteq \{\downarrow, \downarrow^+, \rightarrow, \rightarrow^+\}$ and τ_{com} are sets of binary symbols called, respectively, *navigational* binary symbols and *common* binary symbols. Over such signatures we consider two fragments of first-order logic: FO^2, i.e., the restriction of first-order logic in which only variables x and y are available, and its extension with *counting quantifiers*, C^2, in which quantifiers of the form $\exists^{\geq n}$, $\exists^{\leq n}$, for $n \in \mathbb{N}$ are allowed. We assume that the reader is familiar with their standard semantics. When measuring the length of formulas we assume binary encodings of numbers n in superscripts of quantifiers.

We write $\mathrm{FO}^2[\tau_{bin}]$ or $\mathrm{C}^2[\tau_{bin}]$ where $\tau_{bin} \subseteq \tau_{nav} \cup \tau_{com}$ to denote that the only binary symbols that are allowed in signatures are from τ_{bin}. We will mostly work with two logics: $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$, for τ_{com} being an arbitrary set of common binary symbols, and $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$, i.e., the fragment with counting quantifiers with no common binary symbols.

We are interested in finite unranked, ordered tree structures, in which the interpretation of the symbols from τ_{nav} is fixed: $\downarrow$ is interpreted as the child relation, $\rightarrow$ as the right sibling relation, and $\downarrow^+$ and $\rightarrow^+$ as their respective transitive closures. We read $u \downarrow w$ as "w is a *child* of u" and $u \rightarrow w$ as "w is the *right sibling* of u". We will also use other standard terminology like *ancestor*, *descendant*, *preceding-sibling*, *following-sibling*, etc. The interpretation of symbols from τ_{com} (if present) is not restricted.

We use $x \not\sim y$ to abbreviate the formula stating that x and y are in *free position*, i.e., that they are related by none of the navigational binary predicates available in the signature. Let us call the formulas specifying the relative position of a pair of elements in a tree with respect to binary navigational predicates *order formulas*. There are ten possible order formulas: $x{\downarrow}y$, $y{\downarrow}x$, $x{\downarrow^+}y \wedge \neg(x{\downarrow}y)$, $y{\downarrow^+}x \wedge \neg(y{\downarrow}x)$, $x{\rightarrow}y$, $y{\rightarrow}x$, $x{\rightarrow^+}y \wedge \neg(x{\rightarrow}y)$, $y{\rightarrow^+}x \wedge \neg(y{\rightarrow}x)$, $x \not\sim y$, $x{=}y$. They are denoted, respectively, as: $\theta_{\downarrow}$, $\theta_{\uparrow}$, $\theta_{\downarrow\downarrow^+}$, $\theta_{\uparrow\uparrow^+}$, $\theta_{\rightarrow}$, $\theta_{\leftarrow}$, $\theta_{\Rightarrow^+}$, $\theta_{\Leftarrow^+}$, $\theta_{\not\sim}$, $\theta_{=}$. Let Θ be the set of these ten formulas.

We use symbol $\mathfrak{T}$ (possibly with sub- or superscripts) to denote tree structures. For a given tree $\mathfrak{T}$ we denote by T its universe. A *tree frame* is a tree over a signature containing no unary predicates and no common binary predicates. We will sometimes say that a tree frame $\mathfrak{T}_f$ is the tree frame *of* $\mathfrak{T}$, or that $\mathfrak{T}$ is *based* on $\mathfrak{T}_f$ if $\mathfrak{T}_f$ is obtained from $\mathfrak{T}$ by dropping the interpretation of all unary and common binary symbols. We say that a formula φ is satisfiable over a tree frame if it has a model based on this tree frame.

Given a tree $\mathfrak{T}$, we say that a node $v \in T$ is a *minimal* node (having some fixed property) if there is no $w \in T$ (having this property) such that $\mathfrak{T} \models w{\downarrow^+}v$. A $\downarrow$-path ($\rightarrow$-path) is a sequence of nodes $v_1, \ldots, v_k$ such that $\mathfrak{T} \models v_i{\downarrow}v_{i+1}$ ($\mathfrak{T} \models v_i{\rightarrow}v_{i+1}$), for $i = 1, \ldots, k-1$. Given a $\downarrow$-path ($\rightarrow$-path) P we say that distinct nodes $v_1, \ldots, v_l$ (having some fixed property) are the l *smallest* elements (having this property) on P if for any other $v \in P$ (having this property) we have $\mathfrak{T} \models v_i{\downarrow^+}v$ ($\mathfrak{T} \models v_i{\rightarrow^+}v$) for $i = 1, \ldots, l$. Analogously we define *maximal* and *greatest* elements.

An (atomic) 1-*type* is a maximal satisfiable set of atoms or negated atoms with free variable x. Similarly, an (atomic) 2-*type* is a maximal satisfiable set of atoms or negated atoms with free variables x, y. Note that the numbers of atomic 1- and 2-types are bounded exponentially in the size of the signature. We often identify a type with the conjunction of all its elements. If we work with a signature with empty τ_{com} then 1-types correspond to subsets of τ_0. We denote by $\boldsymbol{\alpha}_\varphi$ the set of 1-types over the signature consisting of symbols appearing in φ.

For a given τ-tree $\mathfrak{T}$, and a node $v \in T$ we say that v *realizes* a 1-type α if α is the unique 1-type such that $\mathfrak{T} \models \alpha[v]$. We denote by $\mathrm{tp}^{\mathfrak{T}}(v)$ the 1-type realized by v. Similarly, for distinct $u, v \in T$, we denote by $\mathrm{tp}^{\mathfrak{T}}(u, v)$ the unique 2-type *realized* by the pair u, v, i.e. the type β such that $\mathfrak{T} \models \beta[u, v]$.

2.2 Normal forms

As usual when working with satisfiability of two-variable logics we employ a Scott-type normal form [26]. We start with its adaptation for the case of $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$.

▶ **Definition 1.** We say that an $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula φ is in *normal form* if

$$\varphi = \forall xy \chi(x, y) \wedge \bigwedge_{i=1}^{m} \forall x(\lambda_i(x) \Rightarrow \exists y(\theta_i(x, y) \wedge \chi_i(x, y))),$$

where $\lambda_i(x)$ is an atomic formula $A(x)$ for some unary symbol A, $\chi(x, y)$ and $\chi_i(x, y)$ are quantifier-free, $\chi_i(x, y)$ do not use symbols from τ_{nav}, and $\theta_i(x, y)$ is an order formula.

We remark that the equality symbol may be used in χ, e.g., we can force a model to contain at most one node satisfying A: $\forall xy(A(x) \wedge A(y) \Rightarrow x{=}y)$. The following lemma can be proved in a standard fashion (cf. e.g., [2]).

▶ **Lemma 2.** *Let φ be an $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula over a signature τ. There exists a polynomially computable $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ normal form formula φ' over signature τ' consisting of τ and some additional unary symbols, such that φ and φ' are satisfiable over the same tree frames.*

Consider a conjunct $\varphi_i = \forall x(\lambda_i(x) \Rightarrow \exists y(\theta_i(x, y) \wedge \chi_i(x, y)))$ of an $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ normal form formula φ. Let $\mathfrak{T} \models \varphi$, and let $v \in T$ be an element such that $\mathfrak{T} \models \lambda_i[v]$. Then an element $w \in T$ such that $\mathfrak{T} \models \theta_i[v, w] \wedge \chi_i[v, w]$ is called a *witness* for v and φ_i. We call w an *upper* witness if $\theta_i(v, w) \models w{\downarrow^+}v$, a *lower* witness if $\theta_i(v, w) \models v{\downarrow^+}w$, a *sibling* witness if $\theta_i(v, w) \models v{\rightarrow^+}w \vee w{\rightarrow^+}v$, and a *free* witness if $\theta_i(v, w) \models v \not\sim w$. We also sometimes simply speak about $\rightarrow^+$-witnesses, $\uparrow$-witnesses, etc.

For C^2 we use a similar but slightly different normal form. One obvious difference is that it uses counting quantifiers, the other is that its $\forall\exists$-conjuncts do not need to contain the θ_i-components, specifying the position of the required witnesses. Refining the normal form by incorporating those components is possible but seems to require an exponential blow-up.

▶ **Definition 3.** We say that a formula $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ is in *normal form*, if:

$$\varphi = \forall x \forall y\ \chi(x, y) \wedge \bigwedge_{i=1}^{m} \left(\forall x\ \exists^{\bowtie_i C_i} y\ \chi_i(x, y)\right),$$

where $\bowtie_i \in \{\leq, \geq\}$, each C_i is a natural number, and $\chi(x, y)$ and all $\chi_i(x, y)$ are quantifier-free.

▶ **Lemma 4** ([13])**.** *Let φ be a formula from $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ over a signature τ. There exists a polynomially computable $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ normal form formula φ' over signature τ' consisting of τ and some additional unary symbols, such that φ and φ' are satisfiable over the same tree frames.*

As in the case of $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ we speak about *witnesses*. Given a normal form $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ formula φ and a tree $\mathfrak{T} \models \varphi$, we say that a node $w \in T$ is a witness for $v \in T$ and a conjunct $\forall x\ \exists^{\bowtie_i C_i} y\ \chi_i(x, y)$ of φ if $\mathfrak{T} \models \chi_i[v, w]$. If additionally $\mathfrak{T} \models w{\downarrow^+}v$ then w is an *upper* witness, if $\mathfrak{T} \models v{\downarrow^+}w$ then w is a *lower* witness, and so on.

In Section 3, when a normal form formula φ is considered we always assume that it is as in Definition 1. In particular we allow ourselves, without explicitly recalling the shape of φ, to refer to its parameter m and components $\chi, \chi_i, \lambda_i, \theta_i$. Analogously, in Section 4 we assume that any normal form φ is as in Definition 3.

3 FO^2 on trees with additional binary relations

In this section we show that the complexity of the satisfiability problem for $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ [2] is retained when the logic is extended with additional, uninterpreted binary relations. Thus we combine here the logic from [2] with the logic from [12]. It means that we need not only to ensure that an element can see realizations of appropriate 1-types in appropriate positions, but also that it is related to them by uninterpreted binary relations in a specific way. In our approach we combine the cutting arguments from [2] with the careful strategy of ensuring witnesses, similar to that from [12] or [27].

▶ **Theorem 5.** *The satisfiability problem for* $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ *over finite trees is* ExpSpace*-complete.*

The lower bound is inherited from $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$. For the upper bound we show that any satisfiable formula φ has a model of depth and degree bounded exponentially in $|\varphi|$. Then we show an auxiliary result allowing us to restrict attention to models in which there is a small number of elements that serve as free witnesses for all elements of the tree. We finally design an alternating exponential time procedure searching for such small models.

3.1 Small model property

Let $\mathfrak{f}$ be a fixed function, which for a given normal form $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula φ returns $96m^3|\boldsymbol{\alpha}_\varphi|^3$. Recall that m is the number of $\forall\exists$-conjuncts of φ and $\boldsymbol{\alpha}_\varphi$ is the set of 1-types over the signature of φ. We will use $\mathfrak{f}$ to estimate the length of paths and the degree of nodes in models. Note that for a given φ the value returned by $\mathfrak{f}$ is exponentially bounded in $|\varphi|$. It should be mentioned that by a more careful analysis one could obtain slightly better bounds (still exponential in $|\varphi|$), but $\mathfrak{f}$ is sufficient for our purposes and allows for a reasonably simple presentation. The following small model property is crucial for obtaining an ExpSpace-upper bound on the complexity of the satisfiability problem. It can be seen as an extension of Theorem 3.3 from [2], where a similar result was proved for FO^2 over trees without additional binary relations.

▶ **Theorem 6** (Small model theorem)**.** *Let φ be a satisfiable normal form formula from* $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$*. Then φ has a model in which the length of every $\downarrow$-path and the degree of each node are bounded exponentially in $|\varphi|$ by $\mathfrak{f}(\varphi)$.*

The proof is split into two lemmas. In the first one we show how to shorten the $\downarrow$-paths and in the second how to reduce the degree of nodes, i.e., to shorten $\rightarrow$-paths.

▶ **Lemma 7.** *Let φ be a normal form* $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ *formula and $\mathfrak{T}$ its model. Then there exists a tree model $\mathfrak{T}'$ for φ whose every $\downarrow$-path has length at most $\mathfrak{f}(\varphi)$.*

Proof. Assume that $\mathfrak{T}$ contains a $\downarrow$-path $P = (v_1, v_2, \ldots, v_n)$ longer than $\mathfrak{f}(\varphi)$. We show that then it is possible to remove some nodes from $\mathfrak{T}$ and obtain a smaller model $\mathfrak{T}_0$. For a node $u \in T$ we define its *projection onto* P as the greatest node $v \in P$, such that $\mathfrak{T} \models v \downarrow^+ u$.

We first distinguish a set W of some relevant elements of $\mathfrak{T}$. W will consist of four disjoint sets W_0, W_1, W_2, W_3. For each 1-type α we mark:

(i) m greatest and m smallest realizations of α on P (or all realizations of α on P if there are less than m of them);

(ii) m realizations of α outside P having greatest projections onto P and m realizations of α outside P having smallest projections onto P (or all realizations of α outside P if there are less than m of them).

Let W_0 be the set consisting of all the marked elements. Let W_1 be a minimal (in the sense of $\subseteq$) set of nodes of $\mathfrak{T}$ such that all the elements from W_0 have all the required witnesses in $W_0 \cup W_1$. Similarly, let W_2 be a minimal set of nodes of $\mathfrak{T}$ such that all the elements from W_1 have all the required witnesses in $W_0 \cup W_1 \cup W_2$. Finally, let W_3 be the set of those projections onto P of elements of $W_0 \cup W_1 \cup W_2$ which are not in $W_0 \cup W_1 \cup W_2$. Let $W := W_0 \cup W_1 \cup W_2 \cup W_3$. To estimate the size of W, observe that $|W_0| \leq 4m|\boldsymbol{\alpha}_\varphi|$, $|W_1| \leq m|W_0|$, $|W_2| \leq m|W_1|$ and $|W_3| \leq |W_0 \cup W_1 \cup W_2|$. Thus $|W| \leq 24m^3|\boldsymbol{\alpha}_\varphi|$.

An *interval* of P *of length* s is a sequence of nodes of the form $(v_i, v_{i+1}, \ldots, v_{i+s-1})$ for some i. We claim that P contains an interval I of length at least $2|\boldsymbol{\alpha}_\varphi|^2 + 2$ having no elements in W. For assume to the contrary that there there is no such interval. Note that the extremal points of P (which are the root and a leaf of $\mathfrak{T}$) are members of W. Hence the points of $W \cap P$ determine at most $|W| - 1$ maximal (possibly empty) intervals not containing elements of W. It follows that $|P| \leq (|W| - 1)(2|\boldsymbol{\alpha}_\varphi|^2 + 1) + |W| < |W|(2|\boldsymbol{\alpha}_\varphi|^2 + 2)$, which by routine calculations gives $|P| < 96m^3|\boldsymbol{\alpha}_\varphi|^3$, a contradiction.

Using the pigeonhole principle we can easily see that in I there are two disjoint pairs of nodes v_k, v_{k+1} and v_l, v_{l+1}, for some $k < l$ such that $\mathrm{tp}^{\mathfrak{T}}(v_{l+i}) = \mathrm{tp}^{\mathfrak{T}}(v_{k+i})$, for $i = 0, 1$. We build a tree $\mathfrak{T}_0$ by replacing in $\mathfrak{T}$ the subtree rooted at v_{k+1} by the subtree rooted at v_{l+1}, setting $\mathrm{tp}^{\mathfrak{T}_0}(v_k, v_{l+1}) := \mathrm{tp}^{\mathfrak{T}}(v_k, v_{k+1})$ and for each v being a sibling of v_{k+1} in $\mathfrak{T}$ setting $\mathrm{tp}^{\mathfrak{T}_0}(v, v_{l+1}) := \mathrm{tp}^{\mathfrak{T}}(v, v_{k+1})$ (all the remaining 2-types are retained from $\mathfrak{T}$). In effect, all the subtrees rooted at elements of P between v_{k+1} and v_l are removed from $\mathfrak{T}$. Note that all elements of W survive our surgery. This guarantees that the elements of $W_0 \cup W_1$ retain all their witnesses. However, some nodes v from $T_0 \setminus (W_0 \cup W_1)$ could lose their witnesses. We can now reconstruct them using the nodes from W_0. This is done by distinguishing several cases. Here we analyse just one of them.

Case 1: $v = v_k$. All the siblings, ancestors and elements in free position to v_k from $\mathfrak{T}$ are retained in $\mathfrak{T}_0$. Thus v_k retains all its sibling, ancestor and free witnesses. There is also no problem with $\downarrow$-witnesses, as v_k retains all its children except v_{k+1}, and v_{k+1} is replaced by v_{l+1} having the same 1-type and connected to v_k exactly as v_{k+1} was. Some $\downarrow\downarrow^+$-witnesses for v_k could be lost however. Let B be a minimal (in the sense of $\subseteq$) set of elements providing the required $\downarrow\downarrow^+$-witnesses for v_k in $\mathfrak{T}$. Note that $|B| \leq m$. Let α be a 1-type realized in B. If all elements of 1-type α from B are in W_0 then there is nothing to do: they survive, and serve as proper $\downarrow\downarrow^+$-witnesses for v_k in $\mathfrak{T}_0$. Otherwise, there must be at least m realizations of α in W_0 (on P or outside P) whose projections onto P in $\mathfrak{T}$ are below v_{l+2}. We can modify the 2-types joining v_k with some of them securing the required $\downarrow\downarrow^+$-witnesses for v_k. This can be done without conflicts, since $v_k \notin W_0 \cup W_1$ and hence it is not required as a witness by any element of W_0.

The remaining cases can be treated similarly.

After the described adjustments all the elements of $\mathfrak{T}_0$ have appropriate witnesses. Since all the 2-types realized in $\mathfrak{T}_0$ are also realized in $\mathfrak{T}$ this ensures that the $\forall\forall$ conjunct of φ is not violated in $\mathfrak{T}_0$. Thus $\mathfrak{T}_0 \models \varphi$.

Note that the number of nodes of $\mathfrak{T}_0$ is strictly smaller than the number of nodes of $\mathfrak{T}$. We can repeat the same shrinking process starting from $\mathfrak{T}_0$, and continue it, obtaining eventually a model $\mathfrak{T}'$ whose paths are bounded as required. ◀

▶ **Lemma 8.** *Let φ be a normal form $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula and $\mathfrak{T} \models \varphi$. Then there exists a model $\mathfrak{T}' \models \varphi$, obtained by removing some subtrees from $\mathfrak{T}$ such that the degree of its every node is bounded by $\mathfrak{f}(\varphi)$.*

Proof. Assume that $\mathfrak{T}$ contains a node v having more than $\mathfrak{f}(\varphi)$ children. We show that then it is possible to remove some of these children together with the subtrees rooted at them and obtain a smaller model $\mathfrak{T}' \models \varphi$. The process is similar to the one described in the proof of Lemma 7. Let $P = (v_1, \ldots, v_k)$ be the $\rightarrow$-path in $\mathfrak{T}$ consisting of all the children of v. We first distinguish a set W of some relevant elements of $\mathfrak{T}$. It will consist of four disjoint sets W_0, W_1, W_2, W_3.

For each 1-type α we mark m greatest and m smallest realizations of α on P (or all realizations of α on P if there are less than m of them). Further we choose $m+1$ elements of P having a realization of α as a descendant (or all such elements if there are less than $m+1$ of them) and for each of them mark one descendant of 1-type α. Let W_0 be the set consisting of all the marked elements. Let W_1 be a minimal set of nodes such that all the elements from W_0 have all the required witnesses in $W_0 \cup W_1$. Similarly, let W_2 be a minimal set of nodes such that all the elements from W_1 have all the required witnesses in $W_0 \cup W_1 \cup W_2$. Finally, let W_3 be the set of those elements of P which are not in $W_0 \cup W_1 \cup W_2$ but have an element from $W_0 \cup W_1 \cup W_2$ in their subtree. Let $W := W_0 \cup W_1 \cup W_2 \cup W_3$. To estimate the size of W, observe that $|W_0| \leq (3m+1)|\boldsymbol{\alpha}_\varphi|$, $|W_1| \leq m|W_0|$, $|W_2| \leq m|W_1|$ $|W_3| \leq |W_0 \cup W_1 \cup W_2|$. Thus, after simple estimations, we have $|W| \leq 24m^3|\boldsymbol{\alpha}_\varphi|$.

An *interval* of P *of length* s is a sequence of nodes of the form $(v_i, v_{i+1}, \ldots, v_{i+s-1})$ for some i. Using arguments similar to those from the proof of Lemma 7 we can show that P contains an interval I with no elements in W, in which there are two disjoint pairs of nodes v_k, v_{k+1} and v_l, v_{l+1}, for some $k < l$ such that $\mathrm{tp}^{\mathfrak{T}}(v_{l+i}) = \mathrm{tp}^{\mathfrak{T}}(v_{k+i})$, for $i = 0, 1$. We build an auxiliary tree $\mathfrak{T}_0$ by removing the subtrees rooted at $v_{k+1}, \ldots, v_l$ and setting $\mathrm{tp}^{\mathfrak{T}_0}(v_k, v_{l+1}) := \mathrm{tp}^{\mathfrak{T}}(v_k, v_{k+1})$ (all the remaining 2-types are retained from $\mathfrak{T}$). Again the elements which lost their witnesses in our construction can regain them by changing their connections to elements from W_0. And again, as in the proof of Lemma 7, the process can be continued until a model with appropriately bounded degree of nodes is obtained. ◀

3.2 Global free witnesses

The small model property from the previous subsection is a crucial step towards an exponential space algorithm for satisfiability. However, it allows for models having doubly exponentially many nodes, which thus cannot be stored in memory. In the case of FO^2 without additional binary relations [2] the corresponding algorithm traversed $\downarrow$-paths guessing for each node v its *full type* storing the sets of 1-types of elements above, below, and in free position to v, similarly to the case of FO^2 with counting from Section 4. Then it took care of *realizing* such full types. This approach would not be sufficient for our current purposes, since the presence of additional binary relations requires us not only to ensure that appropriate 1-types of elements will appear above, below and in free position to a node but also that appropriate 2-types will be realized. This is especially awkward when dealing with free witnesses, since for a given node they are located on different paths. To overcome this difficulty we show that we always can assume that all elements have their free witnesses in a small, exponentially bounded fragment of some model.

▶ **Lemma 9.** *Let φ be a normal form $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula and $\mathfrak{T}$ its model. Let h be the length of the longest $\downarrow$-path in $\mathfrak{T}$ and d the maximal number of $\downarrow$-successors of a node.*

Then there exists a tree $\mathfrak{T}'$ and a set of nodes $F \subseteq T'$, called a global set of free witnesses *such that:*

- *the universes, the 1-types of all elements and the tree frames of $\mathfrak{T}$ and $\mathfrak{T}'$ are identical,*
- $\mathfrak{T}' \models \varphi$,
- *the size of F is bounded by $3(m+1)^3h^2d^2|\boldsymbol{\alpha}_\varphi|$,*
- *F is closed under $\uparrow$, $\leftarrow$ and $\rightarrow$,*
- *for each conjunct of φ of the form $\varphi_i = \forall x(\lambda_i(x) \to \exists y(x \not\sim y \wedge \chi(x,y)))$ and each node $v \in T'$, if $\mathfrak{T}' \models \lambda_i[v]$ then there is a witness for v and φ_i in F.*

Proof. We first describe a procedure which distinguishes in $\mathfrak{T}$ the desired set F. This will contain three disjoint subsets F_0, F_1, F_2. Start with $F_0 = F_1 = F_2 = \emptyset$. For each 1-type α choose $m+1$ maximal elements of type α in $\mathfrak{T}$ (or all of them if there are less than $m+1$ such elements) and make them members of F_0. Close F_0 under $\uparrow$, $\leftarrow$ and $\rightarrow$, i.e., for each member of F_0 add to F_0 also all its ancestors, siblings and all the siblings of its ancestors. This finishes the construction of F_0. Observe that $|F_0| \leq (m+1)hd|\boldsymbol{\alpha}_\varphi|$.

For each $v \in F_0$ and each conjunct of φ of the form $\varphi_i = \forall x(\lambda_i(x) \to \exists y(x \not\sim y \wedge \chi(x,y)))$ if $\mathfrak{T} \models \lambda_i[v]$ and there is no witness for v and φ_i in F_0 then find one in $\mathfrak{T}$ and add it to F_1. Similarly, For each $v \in F_1$ and each conjunct of φ of the form $\varphi_i = \forall x(\lambda_i(x) \to \exists y(x \not\sim y \wedge \chi(x,y)))$ if $\mathfrak{T} \models \lambda_i[v]$ and there is no witness for v and φ_i in $F_0 \cup F_1$ then find one in $\mathfrak{T}$ and add it to F_2.

Take as F the smallest set containing $F_0 \cup F_1 \cup F_2$ and closed under the relations $\uparrow$, $\leftarrow$ and $\rightarrow$. Note that $|F_1| \leq m|F_0| \leq m(m+1)hd|\boldsymbol{\alpha}_\varphi|$, and similarly $|F_2| \leq m|F_1| \leq m^2(m+1)hd|\boldsymbol{\alpha}_\varphi|$. This allows us to estimate the size of F as follows, $|F| \leq (m+1)hd|\boldsymbol{\alpha}_\varphi| + \big(m(m+1)hd|\boldsymbol{\alpha}_\varphi| + m^2(m+1)hd|\boldsymbol{\alpha}_\varphi|\big)hd \leq 3(m+1)^3h^2d^2|\boldsymbol{\alpha}_\varphi|$, as required.

To obtain $\mathfrak{T}'$ we modify some 2-types joining pairs of elements in free position, one of which is in $T \setminus (F_0 \cup F_1)$ and the other in F_0. Consider any element $v \in T \setminus (F_0 \cup F_1)$ and let B be a minimal (with respect to $\subseteq$) set of elements providing the required free witnesses for v in $\mathfrak{T}$. Note that $|B| \leq m$. Let α be a 1-type realized in B. If all elements of 1-type α from B are in F_0 then there is nothing to do: we just retain the connections of v with the elements of type α in F_0. Otherwise there are $m+1$ maximal realizations of α in F_0, and at least m of them is in free position to v. Indeed, $v \notin F_0$ and thus it cannot be an ancestor or a sibling of any of those $m+1$ maximal realizations of α (since F_0 is closed under $\uparrow$, $\leftarrow$ and $\rightarrow$), so if it is not in free position to all then it is a descendant of one of them. But in this case it is in free position to all the other (since maximal realizations of α are in free position to each other). Thus, in this case, for any $w \in B$ of type α we can choose a fresh w' of type α in F_0 in free position to v and set $\mathrm{tp}^{\mathfrak{T}'}(v,w') := \mathrm{tp}^{\mathfrak{T}}(v,w)$. We repeat this step for all 1-types of elements of B, thus ensuring that v has all the required free witnesses in F_0. We repeat this process for all elements of $T \setminus (F_0 \cup F_1)$.

This finishes our construction of $\mathfrak{T}'$. Note that our surgery does not affect the 2-types inside $\mathfrak{T}\restriction(F_0 \cup F_1)$ and the 2-types joining the elements of F_1 with the elements of $T \setminus (F_0 \cup F_1)$. Thus in $\mathfrak{T}'$ all elements of $F_0 \cup F_1$ retain their free witnesses in F and all the remaining elements have appropriate free witnesses in F_0 due to our construction. As we do not change the 2-types joining the elements which are not in free position thus all the upper, lower and sibling witnesses are retained in $\mathfrak{T}'$. Since $\mathfrak{T}'$ realizes only 2-types realized in $\mathfrak{T}$ the universal conjunct of $\forall xy\chi(x,y)$ of φ is satisfied in $\mathfrak{T}'$. Hence, $\mathfrak{T}' \models \varphi$. ◀

3.3 The algorithm

We are now ready to present an alternating algorithm for the finite satisfiability problem for $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$, working in exponential time. Since AExpTime=ExpSpace this justifies Thm. 5. Due to Lemma 2 we can assume that the input formula is given in normal form.

We first sketch our approach. For a given normal form φ the algorithm attempts to build a model $\mathfrak{T} \models \varphi$. It first guesses its fragment $\mathfrak{F}$, of size exponentially bounded in $|\varphi|$, intended to provide free witnesses for all elements of $\mathfrak{T}$, and then expands it down. Namely, it universally chooses one of the leaves v of $\mathfrak{F}$, guesses all its children $w_1, \ldots, w_k$ (at most exponentially many), and guesses 2-types joining w_i-s with all their ancestors, with all elements of $\mathfrak{F}$, and among each other. The algorithm verifies that the guessed elements are consistent with the partial model constructed so far, and if so it universally chooses one of w_i and proceeds with w_i analogously like with v. This process is continued until the algorithm decides that a leaf of $\mathfrak{T}$ is reached.

We must ensure that the structure $\mathfrak{T}$ which is constructed by our algorithm is indeed a model of φ, i.e., all elements of $\mathfrak{T}$ have appropriate witnesses for $\forall\exists$ conjuncts, and that no pair of elements of $\mathfrak{T}$ violates the $\forall\forall$ conjunct. Note that when the algorithm inspects a node v all its siblings and ancestors are present in the memory. This allows to verify that v has the required upper and sibling witnesses. Checking the existence of free witnesses is not problematic too, because, owing to Lemma 9 we assume that they are provided by $\mathfrak{F}$, which is never removed from the memory. Verifying $\downarrow$-witnesses is also straightforward, since we guess all the children $w_1, \ldots, w_k$ of v at once. To deal with $\downarrow^+$-witnesses the algorithm stores some additional data. Namely, together with each w_i it guesses the list of all 2-types (called *promised* 2-*types*) which will be assigned to the pairs consisting of v or its ancestor and a descendant of w_i. This is obviously sufficient to see if v will have the required $\downarrow^+$-witnesses. The algorithm will take care of the consistency of the information about promised types stored in various nodes, and then ensure that all the promised 2-types will indeed be realized.

Turning to the problem of verifying that the universal conjunct of φ is not violated by any pair of elements of $\mathfrak{T}$ note that it is easy for pairs of elements which are not in free position, since at some point during the execution of the algorithm they are both present in the memory and their 2-type is then available. For a pair of elements u_1, u_2 in free position there is an element v such that u_1, u_2 are descendants of two different children of v from the list $w_1, \ldots, w_k$. From information about the promised 2-types guessed together with w_i-s, we can extract the list of 1-types that will appear below each of w_i. Reading this information we see that the 1-types of u_1 and u_2 will appear in free position, and we just need to verify that there is a 2-type consistent with the $\forall\forall$-conjunct which can join them.

A more detailed description of the algorithm together with arguments for its correctness is given in Appendix A.

4 $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ on trees

In this section we prove that the finite satisfiability problem for $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ over trees is ExpSpace-complete. Intuitively, the upper bound proof is a combination of the two proofs from [5] and [7] that solve the problem for FO^2 on trees and for C^2 on linear orders respectively (note that a linear order is just a tree whose each node has at most one child). However, the method in [5] heavily depends on the normal form from Definition 1 where each conjunct corresponds to at most one relative position $\theta \in \Theta$. Although it is possible to bring a $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ formula into an analogous normal form, it seems to require a doubly

exponential blowup (recall that we assume binary coding of the numbers C_i and observe that the number of possible divisions of a set of C_i witnesses into 10 subsets corresponding to 10 order formulas is exponential in C_i). Therefore, to keep the complexity under control, we stay with the usual, less refined normal form from Definition 3, but to compensate it we introduce a new idea combining type information with witness counting.

4.1 Multisets

Any element of a model of a normal form conjunct $\forall x \exists^{\bowtie C_i} y\ \chi$ may require up to C_i witnesses, so we are interested in *multisets* counting these witnesses. To simulate counting up to the value k, we use the function $cut_k : \mathbb{N} \to \{0, 1, 2, \ldots, k, \infty\}$, where $cut_k(i) = i$ for $i \leq k$ and $cut_k(i) = \infty$ otherwise.

Formally, for a given $k \in \mathbb{N}$, a *k-multiset* M of elements from a set S is a function $M : S \to \{0, 1, 2, \ldots, k, \infty\}$. For every element $e \in S$ we interpret $M(e)$, called the *multiplicity* of e in M, as the number of occurrences of e in the multiset M, counted up to k. We employ standard set-theoretic operations, i.e., union $\cup$ and intersection $\cap$ with their natural semantics defined as follows: for given multisets A and B and an arbitrary element e from their domains, we define $(A \cup B)(e) = cut_k\,(A(e) + B(e))$ and $(A \cap B)(e) = \min(A(e), B(e))$. Additionally, we define the empty multiset $\emptyset$ as the function that for any argument returns 0 and the singleton $\{e\}$ of e as the function such that $\{e\}(e) = 1$ and $\{e\}(e') = 0$ for all $e' \neq e$.

4.2 Full types, witness counting and reduced types

We abstract information about nodes in a tree using the following notion.

▶ **Definition 10** (Full type)**.** A *k-full type* $\overline{\alpha}$ (over a signature $\tau = \tau_0 \cup \tau_{nav}$) is a function of type $\overline{\alpha} : \Theta \to \{0, 1, 2, \ldots, k, \infty\}^{2^{\tau_0}}$ (a function which takes a position from Θ and returns a k-multiset of 1-types over τ), that satisfies the following conditions:

- $\overline{\alpha}(\theta_\uparrow)$, $\overline{\alpha}(\theta_\rightarrow)$, $\overline{\alpha}(\theta_\leftarrow)$ is either empty or a singleton,
- $\overline{\alpha}(\theta_=)$ is a singleton, and
- if $\overline{\alpha}(\theta_\uparrow)$ (respectively, $\overline{\alpha}(\theta_\downarrow)$, $\overline{\alpha}(\theta_\rightarrow)$, $\overline{\alpha}(\theta_\leftarrow)$) is empty, then also the multiset $\overline{\alpha}(\theta_{\uparrow\uparrow^+})$ (respectively, $\overline{\alpha}(\theta_{\downarrow\downarrow^+}), \overline{\alpha}(\theta_{\rightrightarrows^+}), \overline{\alpha}(\theta_{\leftleftarrows^+})$) is empty.

Let C be the function that for a given normal form φ (cf. Def. 3) returns $C(\varphi) = \max\{C_i\}_{1 \leq i \leq m}$. We work with k-full types usually in contexts in which a normal form φ is fixed, and we are then particularly interested in $C(\varphi)$-full types. The purpose of a k-full type is to say for a given node v, for each $\theta \in \Theta$ and each 1-type α', how many vertices (counting up to k) of 1-type α' are in position θ to v. Formally:

▶ **Definition 11.** For a given tree $\mathfrak{T}$ and $v \in T$ we denote by $\mathrm{ftp}_k^{\mathfrak{T}}(v)$ the unique k-full type *realized* by v, i.e., the k-full type $\overline{\alpha}$ such that $\overline{\alpha}(\theta_=)$ contains the 1-type of v and for all positions $\theta \in \Theta$ and for all atomic 1-types α' we have that

$$\overline{\alpha}(\theta)(\alpha') = cut_k\left(\#\{w \in T : \mathfrak{T} \models \theta[v, w] \wedge \mathrm{tp}^{\mathfrak{T}}(w) = \alpha'\}\right)$$

where $\#S$ denotes the cardinality of the set S.

We next define functions which for a normal form φ and a $C(\varphi)$-full type $\overline{\alpha}$ say how many witnesses a realization of $\overline{\alpha}$ has for each of the $\forall\exists$ conjuncts of φ (recall that m is the number of such conjuncts) in all possible positions θ.

▶ **Definition 12** (Witness counting functions). Let φ be a normal form formula, and let $\overline{\alpha}$ be a $C(\varphi)$-full type. Assume that $\overline{\alpha}(\theta_{=}) = \{\alpha\}$. We associate with φ and $\overline{\alpha}$ a function $W^{\varphi}_{\overline{\alpha}} : \{1,\ldots,m\} \times \Theta \to \{0,1,\ldots,C(\varphi),\infty\}$, whose values are defined in the following way:

- for $\theta \in \{\theta_{=}, \theta_{\rightarrow}, \theta_{\leftarrow}, \theta_{\downarrow}, \theta_{\uparrow}\}$ and any i:

$$W^{\varphi}_{\overline{\alpha}}(i,\theta) = \begin{cases} 1 & \text{if } \overline{\alpha}(\theta)=\{\alpha'\} \text{ and } \alpha(x)\wedge\alpha'(y)\wedge\theta(x,y)\models\chi_i(x,y) \\ 0 & \text{otherwise,} \end{cases}$$

- for $\theta \in \{\theta_{\leftleftarrows^+}, \theta_{\rightrightarrows^+}, \theta_{\downarrow\downarrow^+}, \theta_{\uparrow\uparrow^+}, \theta_{\nearrow}\}$ and any i:

$$W^{\varphi}_{\overline{\alpha}}(i,\theta) = cut_{C(\varphi)}\left(\sum_{\alpha'\in A_{\alpha,\theta,i}} (\overline{\alpha}(\theta))(\alpha')\right),$$

where $A_{\alpha,\theta,i} = \{\alpha' :\ \alpha(x)\wedge\alpha'(y)\wedge\theta(x,y) \models \chi_i(x,y)\}$.

This way $W^{\varphi}_{\overline{\alpha}}(i,\theta)$ is the number of witnesses (counted up to $C(\varphi)$), in relative position θ, for a node of full type $\overline{\alpha}$ and the formula χ_i from φ.

Now we relate the notion of full types with the satisfaction of normal form formulas.

▶ **Definition 13** (φ-consistency). Let φ be a $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ formula in normal form. Let $\overline{\alpha}$ be a $C(\varphi)$-full type. Assume that $\overline{\alpha}(\theta_{=})$ consists of a 1-type α. We say that $\overline{\alpha}$ is *φ-consistent* if it satisfies the following conditions.

- $\alpha(x) \models \chi(x,x)$,
- $\alpha(x) \wedge \alpha'(y) \wedge \theta(x,y) \models \chi(x,y)$ for all $\theta \in \Theta$ and all $\alpha' \in \overline{\alpha}(\theta)$, and
- for all $1 \leq i \leq m$ the inequality $\sum_{\theta\in\Theta} W^{\varphi}_{\overline{\alpha}}(i,\theta) \bowtie_i C_i$ holds.

Proving the following lemma is routine.

▶ **Lemma 14.** *Assume that a formula $\varphi \in \mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ is in normal form. Then $\mathfrak{T} \models \varphi$ iff every $C(\varphi)$-full type realized in $\mathfrak{T}$ is φ-consistent.*

The next notion will be used to describe information from full types in a (lossy) compressed form. We need this form to obtain tight complexity bounds.

▶ **Definition 15** (φ-reduced type). Let φ be a normal form $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ formula. For a given $C(\varphi)$-full type $\overline{\alpha}$, its φ-reduced form, $\mathrm{rftp}_{\varphi}(\overline{\alpha})$, is the tuple $\left(\alpha, W^{\varphi}_{\overline{\alpha}}, A, B, F\right)$, where $A = \overline{\alpha}(\theta_{\uparrow}) \cup \overline{\alpha}(\theta_{\uparrow\uparrow^+})$, $B = \overline{\alpha}(\theta_{\downarrow}) \cup \overline{\alpha}(\theta_{\downarrow\downarrow^+})$, $F = \overline{\alpha}(\theta_{\rightarrow}) \cup \overline{\alpha}(\theta_{\leftarrow}) \cup \overline{\alpha}(\theta_{\rightrightarrows^+}) \cup \overline{\alpha}(\theta_{\leftleftarrows^+}) \cup \overline{\alpha}(\theta_{\nearrow})$ and $\overline{\alpha}(\theta_{=})$ is the singleton of the 1-type α. If the $C(\varphi)$-full type $\overline{\alpha}$ is realized by a vertex v in $\mathfrak{T}$ then we say that $\mathrm{rftp}_{\varphi}(\overline{\alpha})$ is the φ-reduced type of v. This reduced full type will be denoted also as $\mathrm{rftp}^{\mathfrak{T}}_{\varphi}(v)$.

Intuitively, if a k-full type $\overline{\alpha}$ is realized by a vertex v in a structure $\mathfrak{T}$ then the multisets A, B, F in $\mathrm{rftp}_{\varphi}(\overline{\alpha})$ are respectively the k-multisets of 1-types realized in $\mathfrak{T}$ above, below and in a "non-vertical" position to v.

Let $\overline{\alpha}, \overline{\beta}$ be k-full types. A *combined k-full type* is a k-full type $\overline{\gamma}$, such that $\overline{\gamma}(\theta) = \overline{\alpha}(\theta)$ or $\overline{\gamma}(\theta) = \overline{\beta}(\theta)$ for all positions $\theta \in \Theta$.

▶ **Lemma 16.** *Let $\overline{\alpha}, \overline{\beta}$ be φ-consistent $C(\varphi)$-full types such that their φ-reduced forms are equal. Then the combined $C(\varphi)$-full type $\overline{\gamma}$ of the form $\overline{\gamma}(\theta) = \overline{\alpha}(\theta)$ for $\theta \in \{\theta_{\uparrow}, \theta_{\uparrow\uparrow^+}, \theta_{\rightarrow}, \theta_{\rightrightarrows^+}, \theta_{\nearrow}, \theta_{\leftarrow}, \theta_{\leftleftarrows^+}\}$ and $\overline{\gamma}(\theta) = \overline{\beta}(\theta)$ for $\theta \in \{\theta_{=}, \theta_{\downarrow}, \theta_{\downarrow\downarrow^+}\}$ is also φ-consistent.*

Proof. Obviously $\overline{\gamma}$ satisfies the first two conditions from Definition 13 because $\overline{\alpha}$ and $\overline{\beta}$ do. The third condition is guaranteed by the equality of the witness counting components. ◀

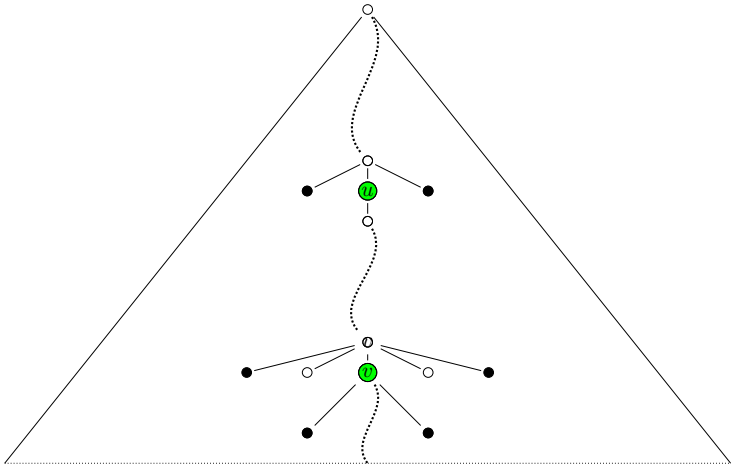


Figure 1 Naive combination of full types.

▶ **Example 17.** Let us observe that in the above lemma the assumption about equality of φ-reduced full types, and in particular their witness counting components, is essential. In [5, Proposition 2] the authors prove that in the setting without counting quantifiers a combined type remains φ-consistent without the assumption about equality of the reduced forms of the original types. The following example shows that in our scenario it is no longer true.

Let φ be a formula saying that every green vertex has at most three direct black neighbors below, on the left or on the right; formally φ is defined as

$$\forall x \exists^{\leq 3} y \ (\mathit{green}(x) \Rightarrow (\mathit{black}(y) \wedge (x \downarrow y \vee x \rightarrow y \vee y \rightarrow x))).$$

Let $\mathfrak{T}$ be a tree model from Fig. 1. Denote $\overline{\alpha} = \mathrm{ftp}^{\mathfrak{T}}_{C(\varphi)}(u)$ and $\overline{\beta} = \mathrm{ftp}^{\mathfrak{T}}_{C(\varphi)}(v)$. Because $\mathfrak{T} \models \varphi$, the $C(\varphi)$-full types $\overline{\alpha}$ and $\overline{\beta}$ are φ-consistent. However the combined $C(\varphi)$-full type $\overline{\gamma}$, in form described in Lemma 16, is not φ-consistent (the black nodes appear in $\overline{\gamma}$ at positions $\theta_{\downarrow}$, $\theta_{\leftarrow}$, $\theta_{\rightarrow}$ four times in total).

4.3 Small model theorem

The general scheme of the decidability proof of finite satisfiability of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ is similar to the one from Section 3. Namely, we demonstrate the small-model property of the logic, showing that every satisfiable formula φ has a tree model of depth and degree bounded exponentially in $|\varphi|$. It is also obtained in a similar way, by first shortening $\downarrow$-paths and then shortening the $\rightarrow$-paths. The technical details differ however.

Recall that given a normal form φ we denote by m the number of its $\forall\exists$ conjuncts, and by $\boldsymbol{\alpha}_\varphi$ the set of 1-types over the signature consisting of the symbols appearing in φ.

▶ **Theorem 18** (Small model theorem). *Let φ be a formula of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in normal form. If φ is satisfiable then it has a a tree model in which every path has length bounded by $3 \cdot (C(\varphi)+2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2$ and every vertex has degree bounded by $(4C(\varphi)^2 + 8C(\varphi)) \cdot |\boldsymbol{\alpha}_\varphi|^5$.*

We split the proof of this theorem into two parts. First, in Lemmas 19 and 20, we show how to reduce the length of paths in a tree and then, in Lemma 21, we show how to reduce the degree of every vertex. We skip most of the details of the proofs due to the space limit.

▶ **Lemma 19** (Cutting lemma). *Let $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ be a formula in normal form and $\mathfrak{T}$ be its model. If there are two vertices $u, v \in T$, such that v is below u and $\mathrm{rftp}^{\mathfrak{T}}_{\varphi}(u) = \mathrm{rftp}^{\mathfrak{T}}_{\varphi}(v)$,*

then the tree $\mathfrak{T}'$, obtained by replacing the subtree rooted at u by the subtree rooted at v, is also a model of φ.

To show this we observe that the $C(\varphi)$-full type of u in tree $\mathfrak{T}'$ is a combination of the $C(\varphi)$-full types of u and v in $\mathfrak{T}$ and thus, by Lemma 16, it is φ-consistent. Then we show that for every other vertex w in $\mathfrak{T}'$ we have $\mathrm{ftp}^{\mathfrak{T}}_{C(\varphi)}(w) = \mathrm{ftp}^{\mathfrak{T}'}_{C(\varphi)}(w)$. Then Lemma 14 guarantees that the obtained tree $\mathfrak{T}'$ is indeed a model of φ.

▶ **Lemma 20.** *Let φ be a satisfiable formula of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in normal form. Then there exists a tree model of φ whose every $\downarrow$-path has length bounded by $3 \cdot (C(\varphi)+2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2$.*

Proof. According to Lemma 19 we can restrict attention to models with the property that every φ-reduced full type appears only once on every $\downarrow$-path. Let $\mathfrak{T} \models \varphi$ be a tree model with this property. Let $v_1, v_2, \ldots, v_n$ be a $\downarrow$-path in $\mathfrak{T}$. Observe that the φ-reduced full types on this path behave in a monotonic way in the sense that for every i and the φ-reduced full types of the $i, (i{+}1)$-th vertices $R_i = (\alpha_i, W_i, A_i, B_i, F_i)$ and $R_{i+1} = (\alpha_{i+1}, W_{i+1}, A_{i+1}, B_{i+1}, F_{i+1})$, we have $A_i \subseteq A_{i+1}, B_{i+1} \subseteq B_i$ and $F_i \subseteq F_{i+1}$. A 1-type α can occur in a multiset from 0 to $C(\varphi)$ times. If α appears more than $C(\varphi)$ times, its multiplicity is ∞. Hence the number of modifications of each multiset from A, B, F is bounded by $(C(\varphi)+2) \cdot |\boldsymbol{\alpha}_\varphi|$. There are up to $|\boldsymbol{\alpha}_\varphi| \cdot (C(\varphi)+2)^{10m}$ φ-reduced full types with fixed multisets A, B, F (because it is the number of all possible 1-types multiplied by the number of all possible witness-counting functions). Combination of these two observations gives us the desired estimation $(C(\varphi)+2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2 \cdot 3$. ◀

▶ **Lemma 21.** *Let φ be a formula in normal form of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ satisfied in a finite tree $\mathfrak{T}$. Then there exists a tree model of φ, obtained by removing some subtrees from $\mathfrak{T}$, such that the degree of every vertex is bounded by $(4C(\varphi)^2 + 8C(\varphi)) \cdot |\boldsymbol{\alpha}_\varphi|^5$.*

To prove this lemma, we first limit the degree of a single vertex. Given a vertex v from T, we mark a small number of children of v as important vertices. Marked vertices are then used as required witnesses for v. Then the reasoning is similar to that of Lemma 19. We introduce an appropriate notion of type and remove all nodes on the horizontal path between two children of the same type, provided that the path does not contain any marked vertex. By repeating this procedure as long as there are vertices of high degree we obtain a desired model of φ.

4.4 Algorithm

In this section we design an algorithm checking if a given formula $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ has a finite tree model. First, by Lemma 4, we can assume that φ is in normal form. Second, by Theorem 18, we can restrict attention to models with exponentially bounded vertex degree and $\downarrow$-path length. The algorithm works in alternating exponential time. The idea of the algorithm is quite simple (see Procedure 4.2 below). For each vertex v we will guess its $C(\varphi)$-full type and check if it is φ-consistent. If it is, we guess the v's children and their full types. After that, we check if their $C(\varphi)$-full types are locally consistent, i.e., if the guessed types coincide with the types realized in the constructed model (see Procedure 4.1). The algorithm starts with $v =$ root and works recursively with its children. The procedure is an adaptation of the one from [5] used in the context of FO^2 without counting quantifiers.

Let us now sketch the arguments for the correctness of Procedure 4.2.

▶ **Lemma 22.** *Procedure 4.2 accepts its input φ iff φ is satisfiable.*

Procedure 4.1 Checking if given $C(\varphi)$-full types are locally-consistent

Input: $C(\varphi)$-full types $\overline{\alpha}$, $\overline{\alpha_1}$, …, $\overline{\alpha_k}$

1: **Return True** if all of the statements below are true. **Return False** otherwise.
2: $\overline{\alpha_i}(\theta_{\leftarrow}) = \overline{\alpha_{i-1}}(\theta_{=})$ for $i > 1$ and $\overline{\alpha_1}(\theta_{\leftarrow}) = \emptyset$
3: $\overline{\alpha_i}(\theta_{\Leftarrow^+}) = \overline{\alpha_{i-1}}(\theta_{\leftarrow}) \cup \overline{\alpha_{i-1}}(\theta_{\Leftarrow^+})$ for $i > 1$ and $\overline{\alpha_1}(\theta_{\Leftarrow^+}) = \emptyset$
4: $\overline{\alpha_i}(\theta_{\rightarrow}) = \overline{\alpha_{i+1}}(\theta_{=})$ for $i < k$ and $\overline{\alpha_k}(\theta_{\rightarrow}) = \emptyset$
5: $\overline{\alpha_i}(\theta_{\Rightarrow^+}) = \overline{\alpha_{i+1}}(\theta_{\rightarrow}) \cup \overline{\alpha_{i+1}}(\theta_{\Rightarrow^+})$ for $i < k$ and $\overline{\alpha_k}(\theta_{\Rightarrow^+}) = \emptyset$
6: $\overline{\alpha}(\theta_{\downarrow}) = \bigcup_{j=1}^{k} \overline{\alpha_j}(\theta_{=})$
7: $\overline{\alpha}(\theta_{\downarrow\downarrow^+}) = \bigcup_{i=1}^{k} \left(\overline{\alpha_i}(\theta_{\downarrow}) \cup \overline{\alpha_i}(\theta_{\downarrow\downarrow^+})\right)$
8: for $1 \leq i \leq k$: $\overline{\alpha_i}(\theta_{\uparrow}) = \overline{\alpha}(\theta_{=})$
9: for $1 \leq i \leq k$:
$\overline{\alpha_i}(\theta_{\not=}) = \overline{\alpha}(\theta_{\not=}) \cup \overline{\alpha}(\theta_{\leftarrow}) \cup \overline{\alpha}(\theta_{\rightarrow}) \cup \overline{\alpha}(\theta_{\Leftarrow^+}) \cup \overline{\alpha}(\theta_{\Rightarrow^+}) \cup \bigcup_{j \neq i} \left(\overline{\alpha_j}(\theta_{\downarrow}) \cup \overline{\alpha_j}(\theta_{\downarrow\downarrow^+})\right)$

Procedure 4.2 Satisfiability test for $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$

Input: Formula $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in normal form.

1: Let MaxDepth $:= 3 \cdot (C(\varphi) + 2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2$
2: Let MaxDeg $:= (4C(\varphi)^2 + 8C(\varphi)) \cdot |\boldsymbol{\alpha}_\varphi|^5$
3: Lvl $:= 0$.
4: **guess** a $C(\varphi)$-full type $\overline{\alpha}$ s.t. $\overline{\alpha}(\theta) = \emptyset$ for all $\theta \in \{\theta_{\uparrow}, \theta_{\uparrow\uparrow^+}, \theta_{\rightarrow}, \theta_{\leftarrow}, \theta_{\Rightarrow^+}, \theta_{\Leftarrow^+}, \theta_{\not=}\}$.
5: **while** Lvl $<$ MaxDepth **do**
6: **if** $\overline{\alpha}$ is not φ-consistent **then reject**
7: **if** $\overline{\alpha}(\theta_{\downarrow}) = \overline{\alpha}(\theta_{\downarrow\downarrow^+}) = \emptyset$ **then accept**
8: **guess** an integer $1 \leq k \leq$ MaxDeg
9: **guess** $C(\varphi)$-full types $\overline{\alpha_1}$, $\overline{\alpha_2}$, …, $\overline{\alpha_k}$
10: **if not** locally-consistent$(\overline{\alpha}, \overline{\alpha_1}, \overline{\alpha_2}, \ldots, \overline{\alpha_k})$ **then reject**
11: Lvl := Lvl + 1
12: **universally choose** $1 \leq i \leq k$; let $\overline{\alpha} = \overline{\alpha_i}$
13: **reject**

Proof. Assume φ is satisfiable. Then there exists a small tree model $\mathfrak{T}$ as guaranteed by Theorem 18. We can run the algorithm and guess exactly the same $C(\varphi)$-full types as in $\mathfrak{T}$. The guessed $C(\varphi)$-full types are locally-consistent and φ-consistent, so Procedure 4.2 accepts.

Assume that Procedure 4.2 accepts its input φ. Then we can reconstruct the tree $\mathfrak{T}$ from the received $C(\varphi)$-full types. The guessed $C(\varphi)$-full types are φ-consistent, which guarantees that we have the right number of witnesses to satisfy the formula. Moreover, the function locally-consistent ensures that the $C(\varphi)$-full types realized in $\mathfrak{T}$ are indeed as we guessed. By Lemma 14, $\mathfrak{T}$ is a tree model for φ and thus φ is satisfiable. ◀

As AExpTime=ExpSpace, and the corresponding lower bound follows from [2] we can conclude this section with the following result.

▶ **Theorem 23.** *The satisfiability problem for* $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ *over finite trees is* ExpSpace*-complete.*

5 Expressive power

A natural question is whether adding counting quantifiers increases the expressive power of two-variable logic over trees. We answer this question concentrating on the classical scenario assuming that signatures contain no common binary symbols. Under this scenario $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ is known to be expressively equivalent to the navigational core of XPath [20]. Here we show that $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ shares the same expressivity. However, it is the presence of the sibling relations which makes FO^2 and C^2 equivalent. Indeed, over unordered trees FO^2 cannot count:

▶ **Theorem 24.** $\mathrm{FO}^2[\downarrow,\downarrow^+]$ *is less expressive than* $\mathrm{C}^2[\downarrow,\downarrow^+]$.

Proof. Let us assume that the signature contains no unary predicates and for $i \in \mathbb{N}$ let $\mathfrak{T}_i$ denote the tree consisting just of a root and its i children. Obviously $\mathfrak{T}_3 \models \exists x \exists^{\geq 3} y\ x{\downarrow^+}y$ while $\mathfrak{T}_2 \not\models \exists x \exists^{\geq 3} y\ x{\downarrow^+}y$. On the other hand, $\mathfrak{T}_2$ and $\mathfrak{T}_3$ are indistinguishable in $\mathrm{FO}^2[\downarrow,\downarrow^+]$. This can be seen by observing that Duplicator has a simple winning strategy in the standard two-pebble game of any length played on $\mathfrak{T}_2$ and $\mathfrak{T}_3$. ◀

Now we turn to the case of full navigational signature.

▶ **Theorem 25.** $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ *and* $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ *are expressively equivalent.*

The core of the proof is the following technical result, allowing us to translate formulas of a simple shape.

▶ **Lemma 26.** *Let φ be a formula of shape $\exists^{\geq k} y(\theta(x,y) \wedge \psi(y))$, where θ is an order formula, ψ is an $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+,\tau_{com}]$ formula with at most one free variable y. Then there exists an $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ formula $trans(\varphi)$ such that for any tree $\mathfrak{T}$, and any $v \in T$ we have $\mathfrak{T} \models \varphi[v]$ iff $\mathfrak{T} \models trans(\varphi)[v]$.*

This lemma is proved by induction on k. E.g., $trans(\exists^{\geq k} y((y{\downarrow^+}x \wedge \neg y{\downarrow}x) \wedge \psi(y)))$ can be simply defined as $\exists y((y{\downarrow^+}x \wedge \neg y{\downarrow}x) \wedge \psi(y) \wedge trans(\exists^{\geq k-1} x(x{\downarrow^+}y \wedge \psi(x))))$. Note that $x{\downarrow^+}y$ is not an order formula and indeed, to make the induction work we need to formulate the thesis for a wider class of possible specifications of the related position of x and y, including not only the order formulas, but also some simple navigational atoms and, more importantly, even some much more complicated descriptions of the relation between x and y. Intuitively, this allows us always to say where the "first" witness is and how the remaining $k-1$ witnesses are related to it.

Lemma 26 can be then generalized to arbitrary formulas with one free variable. This gives Theorem 25. We skip the details due to the space limit.

6 Combining the two extensions

We have proved that two extensions of two-variable logic on trees: the extension with counting quantifiers, $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$, and the extension with additional uninterpreted binary relations, $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+,\tau_{com}]$, remain decidable and retain ExpSpace-complexity of $\mathrm{FO}^2[\rightarrow,\rightarrow^+,\downarrow,\downarrow^+]$. It is tempting to combine both variants into a single logic, i.e., to consider $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+,\tau_{com}]$, the two-variable logic with counting quantifiers and additional binary relation over trees. However, this turns out to lead to a very difficult formalism. Namely, we can reduce to it the long standing open problem of checking non-emptiness of vector addition tree automata.

▶ **Theorem 27.** *The satisfiability problem for* $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ *is at least as hard as checking non-emptiness of vector addition tree automata.*

Proof. To prove the theorem we can mimic the reduction of vector addition tree automata to two-variable logic on *data trees* given in Thm. 4.1 in [4]. Data trees are just trees with an additional, uninterpreted equivalence relation on nodes. In the reduction there the intended equivalence classes are of size at most two. We can easily simulate this by using a common binary symbol $E \in \tau_{com}$, constraining it to be reflexive and symmetric (which is naturally expressible in FO^2), and using counting quantifiers to force each element to be connected by E to at most one other element. The remaining details of the proof remain unchanged. In the proof we do not need to use $\rightarrow$ nor $\rightarrow^+$. ◀

References

1 Bartosz Bednarczyk, Witold Charatonik, and Emanuel Kieronski. Extending two-variable logic on trees. *CoRR*, abs/1611.02112, 2016. URL: `http://arxiv.org/abs/1611.02112`.

2 Saguy Benaim, Michael Benedikt, Witold Charatonik, Emanuel Kieroński, Rastislav Lenhardt, Filip Mazowiecki, and James Worrell. Complexity of two-variable logic on finite trees. *ACM Transactions on Computational Logic*, 17(4):32:1–32:38, 2017. Extended abstract in ICALP 2013.

3 M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4):27, 2011.

4 M. Bojanczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3), 2009.

5 Witold Charatonik, Emanuel Kieronski, and Filip Mazowiecki. Satisfiability of the two-variable fragment of first-order logic over trees. *CoRR*, abs/1304.7204, 2013.

6 Witold Charatonik, Emanuel Kieroński, and Filip Mazowiecki. Decidability of weak logics with deterministic transitive closure. In *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*, page 29, 2014. `doi:10.1145/2603088.2603134`.

7 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and a linear order. *Logical Methods in Computer Science*, 12(2), 2016. `doi:10.2168/LMCS-12(2:8)2016`.

8 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. *ACM Transactions on Computational Logic*, 17(4):31:1–31:27, 2017. Extended abstract in LICS 2013.

9 Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. Vector addition tree automata. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 64–73. IEEE Computer Society, 2004. `doi:10.1109/LICS.2004.1319601`.

10 Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002. `doi:10.1006/inco.2001.2953`.

11 Diego Figueira. Satisfiability for two-variable logic with two successor relations on finite linear orders. *Computing Research Repository*, abs/1204.2495, 2012.

12 E. Grädel, P. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

13 E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *LICS 1997, Proceedings*, pages 306–317, 1997.

14 Emanuel Kieroński. Results on the guarded fragment with equivalence or transitive relations. In C.-H. Luke Ong, editor, *CSL*, volume 3634 of *Lecture Notes in Computer Science*, pages 309–324. Springer, 2005. doi:10.1007/11538363_22.

15 Emanuel Kieroński. Decidability issues for two-variable logics with several linear orders. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*, pages 337–351. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.CSL.2011.337.

16 Emanuel Kieroński, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. In *LICS*, pages 431–440. IEEE Computer Society, 2012. doi:10.1109/LICS.2012.53.

17 Emanuel Kieroński and Martin Otto. Small substructures and decidability issues for first-order logic with two variables. In *LICS*, pages 448–457. IEEE Computer Society, 2005. doi:10.1109/LICS.2005.49.

18 Emanuel Kieroński and Lidia Tendera. On finite satisfiability of two-variable first-order logic with equivalence relations. In *LICS*, pages 123–132. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.39.

19 Amaldev Manuel. Two variables and two successors. In Petr Hlinený and Antonín Kucera, editors, *MFCS*, volume 6281 of *Lecture Notes in Computer Science*, pages 513–524. Springer, 2010. doi:10.1007/978-3-642-15155-2_45.

20 Maarten Marx and Maarten de Rijke. Semantic characterization of navigational XPath. In *First Twente Data Management Workshop (TDM 2004) on XML Databases and Information Retrieval*, pages 73–79, 2004.

21 Martin Otto. Two variable first-order logic over ordered domains. *J. Symb. Log.*, 66(2):685–702, 2001.

22 Leszek Pacholski, Wieslaw Szwast, and Lidia Tendera. Complexity of two-variable logic with counting. In *LICS*, pages 318–327, 1997. doi:10.1109/LICS.1997.614958.

23 I. Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.

24 I. Pratt-Hartmann. Logics with counting and equivalence. In *CSL-LICS 2014, Proceedings*, page 76, 2014.

25 Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations – (extended abstract). In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 2010. doi:10.1007/978-3-642-15205-4_38.

26 Dana Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:477, 1962.

27 Thomas Zeume and Frederik Harwath. Order-invariance of two-variable logic is decidable. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS'16, pages 807–816, New York, NY, USA, 2016. ACM. doi:10.1145/2933575.2933594.

A Algorithm for two-variable logic over trees

In this section we give a more detailed description of the algorithm solving the satisfiability problem for $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$. Then we explain that it has the desired complexity and sketch the argument for its correctness.

The algorithm employs a data structure, storing for each node v the following three components:

- v.`1tp` – the 1-type of v,
- v.`2tp()` – the function which for each w being a sibling of v, an ancestor of v or a member of F, returns the 2-type of (v, w),

Procedure A.1 $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$-sat-test

Input: a formula φ in $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ normal form

guess a tree $\mathfrak{F}$ of depth and degree of nodes bounded by $\mathfrak{f}(\varphi)$ and the number of nodes bounded by $3(m{+}1)^3(\mathfrak{f}(\varphi))^4|\boldsymbol{\alpha}_\varphi|$
for each $v \in F$ **do**
  if v is not a leaf in $\mathfrak{F}$ **then**
    if not *consistent-with-ancestors-siblings-F*(v) **then reject**
    if not *has-upper-sibling-free-witnesses*(v) **then reject**
    Let $w_1, \ldots, w_k$ be the list of the children of v
    if not *ensure-lower-witnesses*$(v, w_1, \ldots, w_k)$ **then reject**
    if not *propagates-promised-2-types*$(v, w_1, \ldots, w_k)$ **then reject**
    if not *respects-universal-conjunct*$(v, w_1, \ldots, w_k)$ **then reject**
universally choose a leaf v of $\mathfrak{F}$; let l be the depth of v in $\mathfrak{F}$
while $l \leq \mathfrak{f}(\varphi)$ **do**
  if not *consistent-with-ancestors-siblings-F*(v) **then reject**
  if not *has-upper-sibling-free-witnesses*(v) **then reject**
  guess a list $w_1, \ldots, w_k$ of children of v; **if** $k > \mathfrak{f}(\varphi)$ **then reject**
  if not *ensure-lower-witnesses*$(v, w_1, \ldots, w_k)$ **then reject**
  if not *propagates-promised-2-types*$(v, w_1, \ldots, w_k)$ **then reject**
  if not *respects-universal-conjunct*$(v, w_1, \ldots, w_k)$ **then reject**
  if $k = 0$ **then accept** *% v is a leaf*
  universally choose $1 \leq j \leq k$ and set $v := w_j$
reject

- v.`p2tp()` – a function which for each ancestor w of v returns a list of promised 2-types, intended to contain all the 2-types which will be realized by w with descendants of v.

We assume that if a node v is guessed during the execution of our procedure then all the above components are constructed.

To avoid presentational clutter in the description of our algorithm we omit some natural conditions on 2-types guessed during its execution, always assuming that they contain the intended navigational atoms, i.e., the 2-type joining an element with its child contains $x{\downarrow}y$, with its right sibling $x{\rightarrow}y$, and so on.

The algorithm is presented as a main Procedure A.1 employing five auxiliary Functions (A.2, A.3, A.4, A.5, A.6).

Function A.2 checks if all guessed components of v are consistent with the information about v's siblings, its ancestors and all the elements of the substructure $\mathfrak{F}$ of global free witnesses.

The next function A.3 checks if v has the required upper, sibling and free witnesses.

Function A.4 checks if the guess of the v's children $w_1, \ldots, w_k$ guarantees lower witnesses for v.

Function A.5 checks if the guess of v.`p2tp()` is propagated to the children of v and consistent with w_i.`p2tp()`.

The last function A.6 checks if the 2-types formed by v with all elements of the constructed model (existing or promised) respect the $\forall\forall$ conjunct.

Using the introduced functions we build our main procedure $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$-`sat-test` (Procedure A.1). We now sketch the arguments showing that it has the advertised complexity and returns correct results.

Function A.2 *consistent-with-ancestors-siblings-F(v)*

for each w being a sibling of v **do**
  let $\beta = v.\mathtt{2tp}(w)$
  if $w.\mathtt{2tp}(v) \neq \beta^{-1}$ **then return false**
if $v \in F$ **then**
  for each $w \in F$ **do**
    let $\beta = v.\mathtt{2tp}(w)$
    if $w.\mathtt{2tp}(v) \neq \beta^{-1}$ **then return false**
if v is the root **then return true**
let u be the father of v
for each w being an ancestor of u **do**
  if $w.\mathtt{2tp}(v) \notin u.\mathtt{p2tp}(w)$ **then return false**
return true

Function A.3 *has-upper-sibling-free-witnesses(v)*

for each conjunct $\forall x(\lambda_i(x) \to \exists y(\theta_i(x,y) \wedge \chi_i(x,y)))$ of φ
with $\theta_i \in \{\theta_\downarrow, \theta_{\downarrow\downarrow^+}, \theta_\rightarrow, \theta_\leftarrow, \theta_{\rightarrow^+}, \theta_{\leftarrow^+}, \theta_{\not\sim}\}$ **do**
  if $v.\mathtt{1tp} \models \lambda_i(x)$ and there is no element w being an ancestor or a sibling of v or a member of F such that $v.\mathrm{2tp}(w) \models \theta_i(x,y) \wedge \chi_i(x,y)$ **then return** false
return true

Function A.4 *ensure-lower-witnesses*$(v, w_1, \ldots, w_k)$

for each conjunct $\forall x(\lambda_i(x) \to \exists y\, \theta_\downarrow(x,y) \wedge \chi_i(x,y))$ of φ **do**
  if $v.\mathtt{1tp} \models \lambda_i(x)$ and there is no w_i such that $v.\mathtt{2tp}(w_i) \models \chi_i(x,y)$ **then**
    return false
for each conjunct $\forall x(\lambda_i(x) \to \exists y\, \theta_{\downarrow\downarrow^+}(x,y) \wedge \chi_i(x,y))$ of φ **do**
  if $v.\mathtt{1tp} \models \lambda_i(x)$ and there is no w_i such that for some $\beta \in w_i.\mathtt{p2tp}(v) : \beta \models \chi_i(x,y)$ **then return false**
return true

Function A.5 *propagates-promised-2-types*$(v, w_1, \ldots, w_k)$

for each u being an ancestor of v **do**
  if $v.\mathtt{p2tp}(u) \neq \bigcup_{i=1}^{k} \left(\{w_i.\mathtt{2tp}(u)^{-1}\} \cup w_i.\mathtt{p2tp}(u)\right)$ **then return false**
return true

▶ **Lemma 28.** *The procedure* FO$^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$-`sat-test` *works in alternating exponential time.*

Proof. During its execution the algorithm guesses $\mathfrak{F}$, and builds a single path P in $\mathfrak{T}$ together with the siblings of the elements from P. The size of $\mathfrak{F}$ is bounded by $3(m+1)^3(\mathfrak{f}(\varphi))^4|\boldsymbol{\alpha}_\varphi|$, the length of P and the degree of nodes are bounded by $\mathfrak{f}(\varphi)$, where m is linear in $|\varphi|$ and $\mathfrak{f}(\varphi)$ and $|\boldsymbol{\alpha}_\varphi|$ are exponential in $|\varphi|$. Thus the algorithm constructs exponentially many nodes. For each node it guesses its 1-type, 2-types joining it with its siblings, ancestors and the elements of $\mathfrak{F}$ (exponentially many in total) and promised 2-types for each of its ancestors (again, information about the 2-types for a single ancestor is bounded exponentially, since the total number of possible 2-types is so bounded). The algorithm makes some consistency

Function A.6 *respects-universal-conjunct*$(v, w_1, \ldots, w_k)$

1: **for each** u being an ancestor of v, a sibling of v, a member of F **do**
2: **if** $v.\mathtt{2tp}(u) \not\models \chi(x,y)$ **then return false**
3: **if** $(v.\mathtt{2tp}(u))^{-1} \not\models \chi(x,y)$ **then return false**
4: **if** v is the root **then return true else** let u be the father of v
5: **for each** w_i **do**
6: let $desc_{w_i} := \{\alpha : \exists \beta \in w_i.\mathtt{p2tp}(u) \wedge \alpha = \beta{\restriction}y\}$.
% $desc_{w_i}$ *is the list of promised* 1*-types of descendants of* w_i
7: **for each** $i \neq j$ **do**
8: **for each** 1-type α from $desc_{w_i}$ **do**
9: **for each** 1-type $\alpha' \in desc_{w_j} \cup \{w_j.\mathtt{1tp}\}$ **do**
10: **if** there is no 2-type β such that $\beta{\restriction}x = \alpha'$ and $\beta{\restriction}y = \alpha$ and $\beta(x,y) \models \theta_{\mathcal{A}}(x,y) \wedge \chi(x,y)$ **then return false**
11: **return** true

and correctness checking, which can be easily done in time polynomial in the size of the guesses. Hence the lemma follows. ◀

▶ **Lemma 29.** *The procedure* $\mathrm{FO}^2[{\downarrow}, {\downarrow^+}, {\rightarrow}, {\rightarrow^+}, \tau_{com}]$-`sat-test` *accepts its input* φ *iff* φ *is satisfiable.*

Proof. (Sketch.) Assume φ has a model. By Theorem 6 it has a model $\mathfrak{T}$ whose depth and degree of nodes are bounded by $\mathfrak{f}(\varphi)$. By Lemma 9 there is a model $\mathfrak{T}'$ based on the same frame as $\mathfrak{T}$, in which one can distinguish a set F, of size at most $3(m+1)^3(\mathfrak{f}(\varphi))^4|\boldsymbol{\alpha}_\varphi|$, providing free witnesses for all elements of $\mathfrak{T}'$. Our algorithm can just take $\mathfrak{F} := \mathfrak{T}'{\restriction}F$ and make all its guesses in accordance with $\mathfrak{T}$.

For the opposite direction assume that our algorithm has an accepting run. From this run we can naturally extract a partially defined tree structure $\mathfrak{T}$ and its substructure $\mathfrak{F}$. $\mathfrak{T}$ has defined its tree frame, 1-types of all nodes ($v.\mathtt{1tp}$ components), 2-types of nodes not in free position and 2-types of nodes in free position at least one of which is in F: the 2-type joining v and w is stored in $v.\mathtt{2tp}(w)$ if v is a descendant of w, or if $w \in F$ and $v \notin F$, and in both $v.\mathtt{2tp}(w)$ and $w.\mathtt{2tp}(v)$ if v and w are siblings or $v, w \in F$. Note that the function *consistent-with-ancestors-siblings-F* ensures that the 2-types can be assigned without conflicts. This function, together with function *propagates-promised-2-types* ensures also the consistency of the information about promised 2-types.

What is missing is 2-types of pairs of elements u_1, u_2 in free position none of which is in F. In this case there is an element v such that u_1, u_2 are descendants of two different children of v from the list $w_1, \ldots, w_k$. Then, due to lines 7-10 of the function *respects-universal-conjunct*, there exists a 2-type consistent with the $\forall\forall$-conjunct which can join them.

The constructed tree $\mathfrak{T}$ is indeed a model of φ: *respects-universal-conjunct* takes care of $\forall\forall$ constraint of φ, the sibling, upper and free witnesses are ensured due to function *has-upper-sibling-free-witnesses* and lower witnesses are guaranteed by function *ensure-lower-witnesses* which uses the information about promised-2-types. ◀

On the (In)Succinctness of Muller Automata*

Udi Boker

Interdisciplinary Center (IDC), Herzliya, Israel

Abstract

There are several types of finite automata on infinite words, differing in their acceptance conditions. As each type has its own advantages, there is an extensive research on the size blowup involved in translating one automaton type to another.

Of special interest is the Muller type, providing the most detailed acceptance condition. It turns out that there is inconsistency and incompleteness in the literature results regarding the translations to and from Muller automata. Considering the automaton size, some results take into account, in addition to the number of states, the alphabet length and the number of transitions while ignoring the length of the acceptance condition, whereas other results consider the length of the acceptance condition while ignoring the two other parameters.

We establish a full picture of the translations to and from Muller automata, enhancing known results and adding new ones. Overall, Muller automata can be considered less succinct than parity, Rabin, and Streett automata: translating nondeterministic Muller automata to the other nondeterministic types involves a polynomial size blowup, while the other way round is exponential; translating between the deterministic versions is exponential in both directions; and translating nondeterministic automata of all types to deterministic Muller automata is doubly exponential, as opposed to a single exponent in the translations to the other deterministic types.

1998 ACM Subject Classification D.2.4 Software/Program Verification, F.4 Mathematical Logic and Formal Languages

Keywords and phrases Automata, Omega-regular languages, Determinization

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.12

1 Introduction

Automata on infinite words were introduced in the 60's, in the course of solving fundamental decision problems in mathematics and logic [4, 14, 10, 16]. Today, they are widely used in formal verification and synthesis of nonterminating systems, where their size and the complexity of performing operations on them play a key role. Unlike automata on finite words, there are several types of automata on infinite words, differing in their acceptance conditions, most notably Büchi [4], Muller [14], Rabin [16], Streett [21], and parity [13]. Each of the types has its own advantages, for which reason there is an extensive research on the size blowup involved in the translations between them [10, 17, 18, 15, 22, 5, 19, 2, 20].

The size of an automaton can be generally viewed as the sum (or maximum) of its element sizes, namely the maximum of the alphabet length, the number of states, the number of transitions, and the index, where the latter denotes the size of the acceptance condition. For Büchi automata the index is 1, for parity it can be as large as the number of states, and for Rabin, Streett, and Muller it can be exponentially larger than the number of states.

When analyzing translations between automata, the first measure to consider is the size blowup, while a second consideration is the influence of and on each of the automaton

* This work was supported by the Israel Science Foundation grant 1373/16.

elements. For example, nondeterministic Büchi automata of size n can be translated to deterministic Rabin automata of size in $2^{O(n \log n)}$; The constructed Rabin automata have, by some algorithms, an index in $O(n)$ [17], while by others it is in $2^{O(n)}$ and the number of states is slightly smaller [19].

In addition to the number of states and the index, a central element to consider is the alphabet length, which might be exponentially larger than the number of states. There are cases in which the alphabet length has no influence on the size blowup (e.g. [22]) and others in which it significantly influences it (e.g. [7]). For lower bound results, the aim is to provide a family of languages over a fixed alphabet. Considering the Büchi-determinization example, Michel provided a matching lower bound of $2^{\Omega(n \log n)}$ [11], however he used Büchi automata over alphabets of length linear in the number of states (and therefore with quadratically many transitions). Afterwards, Löding improved the lower bound to be over a fixed alphabet [9]. An often productive approach is to start with a rich alphabet of length exponential in the number of states, get a lower bound with respect to the number of states, while ignoring the alphabet length and the number of transitions, and then enhance it to be over a fixed alphabet [22, 3]. The challenge in this approach is to encode the rich alphabet by a fixed one without enlarging the number of states.

Less central is the number of transitions, which might be as large as the product of the alphabet length and quadratically the number of states. Nevertheless, this element is often taken into account in analyzing the size blowup [8, 19]. Moreover, in recent years there is a growing interest in automata with acceptance labeling on transitions rather than on states (e.g. [5]), further increasing the importance of this element.

We concentrate on the translations to and from Muller automata. It turns out that there is inconsistency and incompleteness in the literature results regarding these translations.

The Muller condition explicitly lists the exact subsets of states that may be visited infinitely often along an accepting run. This is in distinction to other types, such as Rabin and Streett, which specify a list of constraints on the subsets of states that are visited infinitely often. It is therefore reasonable to assume that Rabin and Streett automata can be exponentially more succinct than Muller automata, which is indeed the case [17].

An interesting question is whether the explicit Muller condition can allow for automata that are exponentially more succinct than the other types. The literature answer seems to be yes. In [9], there is an exponential lower bound for the translation of deterministic Streett to deterministic Rabin automata, and vice versa. It is claimed there that these lower bounds also hold for the translations of deterministic Muller to deterministic Rabin and Streett automata. However, a closer look at the family of Streett automata that is used in the lower-bound proof suggests that equivalent deterministic Muller automata need an index exponential in the number of states. Thus, they are not exponentially smaller than the equivalent deterministic Rabin automata. Another candidate family of languages $\{L_n\}$ is the one used in Michel's lower bound [11]. Löding shows that L_n is recognized by a deterministic Muller automaton with only n^2 states, whereas an equivalent deterministic Rabin automaton needs $2^{\Omega(n \log n)}$ states [9]. Yet, these Muller automata also require an index exponential in the number of states.

It is thus open whether the explicit Muller condition can allow for significantly smaller automata than equivalent Rabin and Streett automata. We answer it positively, providing families of deterministic Muller automata of size in $O(n)$, for which we prove that equivalent deterministic Rabin and Streett automata have at least $2^{\Omega(n)}$ states (Theorems 4 and 5). We leave open a gap between this lower bound and the $2^{O(n \log n)}$ upper bound of the State Appearance Records construction.

Table 1 The size blowup involved in the translations of Muller automata to Büchi, Parity, Rabin, and Streett automata.

From \ To	Deterministic			Non-Deterministic			
	P	R	S	B	P	R	S
Det. Muller	$2^{O(n \log n)}$ $2^{\Omega(n)}$ Prop. 3 Thms. 4,5			$\Theta(n^3)$ Prop. 6 Thm. 10		$O(n^3)$ $\Omega(n^2)$ Prop. 6 Thm. 9	$\Theta(n^2)$ Prop. 7 Thm. 8
Non-Det. Muller	$2^{O(n^3 \log n^3)}$ $2^{\Omega(n)}$ Prop. 1,2						

As for the translation of Muller automata to nondeterministic automata of the other types, there is a known construction that involves an $O(n^3)$ size blowup in the translation to Büchi, parity, and Rabin automata, which can be improved to $O(n^2)$ for the translation to Streett. We provide corresponding lower bounds, tight for the translations to Büchi, parity, and Streett, and with a gap between $\Omega(n^2)$ and $O(n^3)$ for the translation to Rabin (Theorems 8-10).

Regarding the translations to Muller automata, Safra shows that the translation of deterministic Büchi to nondeterministic Muller automata involves an exponential size blowup, and of nondeterministic Büchi to deterministic Muller a doubly exponential blowup [17]. Safra does include the index in the automaton size, however uses in the latter lower bound an alphabet of length exponential in the number of states.

We strengthen Safra's lower bounds, by providing families of languages over a fixed alphabet and by showing that the size blowup stems directly from the structure of the translated automata, regardless of the acceptance condition – our languages are recognized by looping automata, which are Büchi automata all of whose states are accepting (Theorems 12 and 16).

For the translation of deterministic automata as well as for the translation to nondeterministic Muller automata, the bounds are tight for all types. Considering the translations of nondeterministic automata to deterministic Muller automata, the bounds are tight for looping, weak, and co-Büchi automata, there is a gap between $2^{2^{\Omega(n)}}$ and $2^{2^{O(n \log n)}}$ for Büchi, and a gap between and $2^{2^{\Omega(n)}}$ and $2^{2^{O(n^2 \log n^2)}}$ for parity, Rabin, and Streett.

The translation blowups are summarized in Tables 1 and 2. Table 1 only includes types to which Muller automata can always be translated, while Table 2 includes additional types, as they all can be translated to Muller. Theorems 4-5 and 8-10 are new, while Theorems 12 and 16 strengthen known results. On the technical level, Theorem 4 has the most involved proof, Theorem 5 is proved analogously, and the proofs of Theorems 8 and 9 provide two different lower-bound techniques, on top of which the proof of Theorem 10 is built. Theorems 12 and 16 strengthen lower-bound results that were known for Büchi automata over a linear or an exponential alphabet to corresponding results for looping automata over a fixed alphabet.

Table 2 The size blowup involved in the translations to Muller automata. "All" stands for the automata types as abbreviated in the table by their first letter, namely Looping, Weak, Co-Büchi, Büchi, Parity, Rabin, and Streett.

<table>
<tr><th colspan="2">From \ To</th><th>Det. Muller</th><th>Non-Det. Muller</th></tr>
<tr><td>Det.</td><td>All</td><td colspan="2">$2^{\Theta(n)}$
Prop. 11, Thm. 12</td></tr>
<tr><td rowspan="7">Non-Det.</td><td>L</td><td rowspan="3">$2^{2^{\Theta(n)}}$
Prop. 13, Thm. 16</td><td rowspan="7">$2^{\Theta(n)}$
Prop. 11
Thm. 12</td></tr>
<tr><td>W</td></tr>
<tr><td>C</td></tr>
<tr><td>B</td><td>$2^{2^{O(n \log n)}}$</td><td rowspan="3">$2^{2^{\Omega(n)}}$
Prop. 14,15
Thm. 16</td></tr>
<tr><td>P</td><td rowspan="3">$2^{2^{O(n^2 \log n^2)}}$</td></tr>
<tr><td>R</td></tr>
<tr><td>S</td></tr>
</table>

2 Preliminaries

Given a finite alphabet Σ, a *word* over Σ is a (possibly infinite) sequence $w = w(0) \cdot w(1) \cdots$ of letters in Σ.

An *automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $Q_0 \subseteq Q$ is a set of initial states, and α is an acceptance condition. The automaton $\mathcal{A}$ may have several initial states and the transition function may specify many possible transitions for each state and letter, and hence we say that $\mathcal{A}$ is *nondeterministic*. In the case where $|Q_0| = 1$ and for every $q \in Q$ and $\sigma \in \Sigma$, we have $|\delta(q, \sigma)| \leq 1$, we say that $\mathcal{A}$ is *deterministic*. For a state q of $\mathcal{A}$, we denote by $\mathcal{A}^q$ the automaton that is derived from $\mathcal{A}$ by changing the set of initial states to $\{q\}$.

A *run*, or a *path*, $r = r(0), r(1), \cdots$ of $\mathcal{A}$ on $w = w(0) \cdot w(1) \cdots \in \Sigma^\omega$ is an infinite sequence of states such that $r(0) \in Q_0$, and for every $i \geq 0$, we have $r(i+1) \in \delta(r(i), w(i))$.

Acceptance is defined with respect to the set $inf(r)$ of states that the run r visits infinitely often. Formally, $inf(r) = \{q \in Q \mid \text{ for infinitely many } i \in \mathbb{N}, \text{ we have } r(i) = q\}$. As Q is finite, it is guaranteed that $inf(r) \neq \emptyset$. The run r is *accepting* iff the set $inf(r)$ satisfies the acceptance condition α, and otherwise it is *rejecting*.

Several acceptance conditions are studied in the literature; the main ones are:

- *Büchi*, where $\alpha \subseteq Q$, and r is accepting iff $inf(r) \cap \alpha \neq \emptyset$.
- *co-Büchi*, where $\alpha \subseteq Q$, and r is accepting iff $inf(r) \cap \alpha = \emptyset$.
- *weak* is a special case of the Büchi condition, where every strongly connected component of the automaton is either contained in α or disjoint to α; that is, no strongly connected component has a state in α and some other state not in α.
- *looping* is a special case of the Büchi condition, where $\alpha = Q$, meaning that all states are accepting.
- *parity*, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_{2k}\}$ with $\alpha_1 \subset \alpha_2 \subset \cdots \subset \alpha_{2k} = Q$, and r is accepting if the minimal index i for which $inf(r) \cap \alpha_i \neq \emptyset$ is even.
- *Rabin*, where $\alpha = \{\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\}$, with $\alpha_i, \beta_i \subseteq Q$ and r is accepting iff for some $1 \leq i \leq k$, we have $inf(r) \cap \alpha_i \neq \emptyset$ and $inf(r) \cap \beta_i = \emptyset$.

- *Streett*, where $\alpha = \{\langle\beta_1, \alpha_1\rangle, \langle\beta_2, \alpha_2\rangle, \ldots, \langle\beta_k, \alpha_k\rangle\}$, with $\beta_i, \alpha_i \subseteq Q$ and r is accepting iff for all $1 \leq i \leq k$, we have $inf(r) \cap \beta_i = \emptyset$ or $inf(r) \cap \alpha_i \neq \emptyset$.
- *Muller*, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, with $\alpha_i \subseteq Q$ and r is accepting iff for some $1 \leq i \leq k$, we have $inf(r) = \alpha_i$.

Notice that Büchi and co-Büchi are special cases of the parity condition, which is in turn a special case of both the Rabin and Streett conditions.

The number of sets in the parity and Muller acceptance conditions or pairs in the Rabin and Streett acceptance conditions is called the *index* of the automaton. For looping, weak, co-Büchi, and Büchi automata, the index is 1.

The *size* of an automaton is the maximum size of its elements; more precisely, it is the maximum of the alphabet length, the number of states, the number of transitions, and the index.

An automaton accepts a word if it has an accepting run on it. The language of an automaton $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We also say that $\mathcal{A}$ *recognizes* the language $L(\mathcal{A})$. Two automata, $\mathcal{A}$ and $\mathcal{A}'$, are *equivalent* iff $L(\mathcal{A}) = L(\mathcal{A}')$.

For a finite path $C = q_1 q_2 \cdots q_n$, we say that C is accepting (resp., rejecting) if the infinite path C^ω is accepting (resp., rejecting). Notice that the union of two Rabin-rejecting (finite) paths is Rabin-rejecting, and of two Streett-accepting (finite) paths is Streett-accepting.

The *class* of an automaton characterizes its branching mode (deterministic or nondeterministic) and its acceptance condition. In the more technical paragraphs, we shall denote the different classes of automata by three letter acronyms in $\{\text{D}, \text{N}\} \times \{\text{L}, \text{W}, \text{B}, \text{C}, \text{P}, \text{R}, \text{S}, \text{M}\} \times \{\text{W}\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance-condition (looping, weak, Büchi, co-Büchi, parity, Rabin, Streett, or Muller); and the third indicates that the automaton runs on words. For example, DBW stands for deterministic Büchi automata on words.

Büchi, parity, Rabin, Streett, and Muller automata have the same expressive power, recognizing all ω-regular languages. Looping, weak, and co-Büchi automata, as well as deterministic Büchi automata, are less expressive.

3 From Muller

We start with the translation of nondeterministic Muller automata to deterministic automata of the other types, for which there are known singly-exponential constructions. We continue, in Section 3.2, with the translation of deterministic Muller to deterministic automata of the other types, for which we answer positively the open question of whether an exponential size blowup is inevitable. In Section 3.3, we provide lower bounds for the known polynomial translations of Muller automata to nondeterministic automata of the other types.

3.1 From Nondeterministic Muller To Deterministic Types

The translation of Muller automata to deterministic automata of the other types involves a singly exponential size blowup. A possible construction is to first translate the Muller automaton into a Büchi automaton, involving an $O(n^3)$ size blowup (Prop. 6), and then determinize it into a Rabin or parity automaton of size in $2^{O(n^3 \log n^3)}$ [17, 15]. An alternative approach is to translate the NMW to a nondeterministic Streett automaton, which only involves a $O(n^2)$ blowup (Prop. 7), and then determinize the latter. Yet, as the determinization of Streett automata is more involved [18, 15], the overall size blowup is not improved.

▶ **Proposition 1** ([17, 15]). *Muller automata of size n can be translated to parity, Rabin, and Streett automata of size in $2^{O(n^3 \log n^3)}$.*

An exponential lower bound for the determinization of all automata types is trivial, by reduction to the case of finite words.

▶ **Proposition 2.** *Determinizing all automata types involves at least an exponential size blowup.*

It may be interesting to close the gap between the trivial $2^{\Omega(n)}$ lower bound and the $2^{O(n^3 \log n^3)}$ upper bound.

3.2 From Deterministic Muller To Deterministic Types

Translating deterministic Muller automata to deterministic parity, Rabin, and Streett automata can be done by the State Appearance Records (SAR) construction [6]. The translation of a DMW with l states results in a DPW with up to $2^{O(l \log l)}$ states, regardless of the DMW's index.

▶ **Proposition 3** ([6]). *Deterministic Muller automata of size n can be translated to deterministic parity, Rabin, and Streett automata of size in $2^{O(n \log n)}$.*

We show below that an exponential size blowup in the translation to parity, Rabin, and Streett automata is inevitable. The proofs borrow ideas from lower bound proofs in [9] and [1], building on the property of the Rabin (resp., Streett) acceptance condition, according to which the union of two rejecting (resp., accepting) cycles is rejecting (resp., accepting).

We start with the translation to Rabin automata. Consider the family of DMWs $\{\mathcal{D}_n\}$, as depicted in Figure 1. The DMW $\mathcal{D}_n$ accepts words over the "alphabet" $[-n..n]$, in which the "letters" that appear infinitely often are exactly all of the "letters" between $-i$ and i, for some $i \in [1..n]$. Technically, a "letter" i is the finite word $a^i\#$ and $-i$ is $b^i\#$.

Let $w_{i,j}$, for $i < j \in [-n..n]$, be words in which exactly all of the letters in $[i..j]$ appear infinitely often. We show that a DRW $\mathcal{A}$ equivalent to $\mathcal{D}_n$ has at least 2^{n-1} states, by proving that a run r of $\mathcal{A}$ on the word $w_{-(i+1),i+1}$ visits at least twice the number of states that a run r_i of $\mathcal{A}$ on $w_{-i,i}$ visits.

The proof idea is intuitively as follows. Let r' and r'' be the runs of $\mathcal{A}$ on $w_{-i,i+1}$ and $w_{-(i+1),i}$, respectively. Then: I) The runs r' and r'' contain the run r_i, thus visit at least as many states as r_i. II) By the definition of $\mathcal{D}_n$, both r' and r'' are rejecting. III) According to the property of the Rabin condition, the union of r' and r'' is rejecting. IV) The runs r' and r'' visit infinitely often disjoint sets of states – if they had a common state, their union would have been a rejecting run of $\mathcal{A}$ on $w_{-(i+1),i+1}$, contradicting its acceptance by $\mathcal{D}_n$. V) The run r contains the runs r' and r'', thus visit at least twice the number of states that r_i visits.

▶ **Theorem 4.** *The translation of deterministic Muller automata to deterministic Rabin automata involves a size blowup of at least $2^{\Omega(n)}$. In particular, there is a family $\{\mathcal{D}_n\}_{n \geq 1}$ of DMWs with $2n+1$ states, $4n$ transitions, and n accepting sets, for which equivalent DRWs have at least 2^{n-1} states.*

Proof. Consider the family $\{\mathcal{D}_n\}$ of DMWs over $\Sigma = \{a, b, \#\}$, as depicted in Figure 1, and let $\mathcal{R}$ be a DRW equivalent to $\mathcal{D}_n$.

For readability, we define for every $i \in [1..n]$, $\widehat{i} = a^i\#$ and $\widehat{-i} = b^i\#$. Then, the language of $\mathcal{D}_n$ can be defined over the finite words $\widehat{j}$, for $j \in [-n..-1] \cup [1..n]$. For simplifying the expressions, we will speak of $\widehat{j}$, for $j \in [-n..n]$, while considering $\widehat{0}$ to be the empty word.

The Deterministic Muller automata $\mathcal{D}_n$, $\mathcal{D}'_n$, and $\mathcal{D}''_n$.

The acceptance conditions:

$\mathcal{D}_n$: The sets Q_i, for $i \in [1..n]$, where $Q_i = \{q_j \mid j \in [-i..i]\}$

$\mathcal{D}'_n$: The sets Q'_i and Q''_i, for $i \in [1..n]$, where $Q'_i = \{q_j \mid j \in [-i..i-1]\}$ and $Q''_i = \{q_j \mid j \in [-(i-1)..i]\}$

$\mathcal{D}''_n$: The sets P_i, for $i \in [0..n]$, where $P_i = \{q_j \mid j \in [-i..n-i]\}$

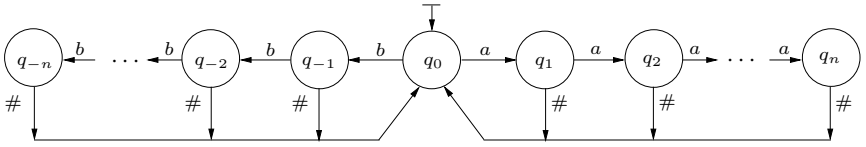


Figure 1 Deterministic Muller automata $\mathcal{D}_n$, $\mathcal{D}'_n$, and $\mathcal{D}''_n$ of size in $O(n)$. For $\mathcal{D}_n$ and $\mathcal{D}'_n$, equivalent deterministic Rabin and Streett automata, respectively, have at least 2^{n-1} states. For $\mathcal{D}''_n$, equivalent nondeterministic Streett automata have at least $n^2/2$ states.

We say that a state q of $\mathcal{R}$ is *significant* if it is reachable after reading a # (namely after an $\widehat{i}$ subword), and from which the automaton can accept some word. Observe that for every significant state q, we have $L(\mathcal{R}^q) = L(\mathcal{R})$, namely the automaton that we get from $\mathcal{R}$ by changing the initial state to q is equivalent to $\mathcal{R}$ (and to $\mathcal{D}_n$). This is the case since every accepting run of $\mathcal{D}_n$ returns to the initial state after reading a #.

We prove by induction on h, for $h \in [1..n]$, the following claim, from which the theorem immediately follows: For every significant state q of $\mathcal{R}$, there exist finite words $u, v \in \Sigma^*$, such that:

(i) For every $j \in [-n..n]$, $\widehat{j}$ appears in u at its start position or after a # iff $j \in [-h..h]$. The same w.r.t. v.

(ii) The run of $\mathcal{R}^q$ on u reaches some significant state p.

(iii) The run of $\mathcal{R}^p$ on v returns to p, while visiting at least 2^{h-1} different significant states.

The base case, for $h = 1$, is trivial, as it means a cycle of size at least 1.

In the induction step, we assume the induction hypothesis for h and prove it for $h{+}1$. We do it in two phases.

Phase 1. We first show that we can add $\widehat{h{+}1}$ as well as $\widehat{-(h{+}1)}$ to u and v, while keeping the induction hypothesis. Formally, we claim that for every significant state q of $\mathcal{R}$, there exist finite words $u', v', u'', v'' \in \Sigma^*$, such that:

(i') For every $j \in [-n..n]$, $\widehat{j}$ appears in u' at its start position or after a # iff $j \in [-h..h{+}1]$. The same w.r.t. v'.

(ii') The run of $\mathcal{R}^q$ on u' reaches some significant state p'.

(iii') The run of $\mathcal{R}^{p'}$ on v' returns to p', while visiting at least 2^{h-1} different significant states.

(i'') For every $j \in [-n..n]$, $\widehat{j}$ appears in u'' at its start position or after a # iff $j \in [-(h{+}1)..h]$. The same w.r.t. v.

(ii'') The run of $\mathcal{R}^q$ on u'' reaches some significant state p''.

(iii'') The run of $\mathcal{R}^{p''}$ on v'' returns to p'', while visiting at least 2^{h-1} different significant states.

We prove below the claim w.r.t. u' and v', while the case of u'' and v'' is completely analogous.

Consider a significant state q, which we also denote by p_0, and let u_0 and v_0 be finite words that satisfy requirements i-iii of the induction hypothesis w.r.t. p_0. We define the

finite word $z_0 = u_0 v_0 \widehat{h+1}$. Notice that a $\widehat{j}$-word appears in z_0 iff $j \in [-h..h+1]$. Let p_1 be the significant state that $\mathcal{R}^{p_0}$ reaches when reading z_0.

We iteratively continue as above, taking words u_1, v_1, and z_1 w.r.t p_1, etc., until reaching an iteration i, for which there is $k < i$, such that $p_i = p_k$.

Observe that the words $u' = z_0 z_1 \cdots z_{k-1}$ and $v' = z_k z_{k+1} \cdots z_{i-1}$ satisfy the requirements i'-iii': they contain a $\widehat{j}$-word iff $j \in [-h..h+1]$; the run of $\mathcal{R}^q$ on u' reaches the significant state p_k; and the run of $\mathcal{R}^{p_k}$ on v' returns to p_k, while visiting at least 2^{h-1} different significant states. The latter holds, since the run of $\mathcal{R}^{p_k}$ on z_k already visits at least 2^{h-1} different significant states.

Phase 2. We continue with showing that the induction claim holds for $h+1$.

Consider a significant state q, denoted by p_0, and let u'_0 and v'_0 be finite words that satisfy requirements i'-iii' w.r.t. q. Let p'_1 be the significant state that $\mathcal{R}^{p_0}$ reaches when reading $u'_0 v'_0$. Now, let u''_0 and v''_0 be finite words that satisfy requirements i"-iii" w.r.t. p'_1. Let p_1 be the significant state that $\mathcal{R}^{p'_1}$ reaches when reading $u''_0 v''_0$. We define the finite word $z_0 = u'_0 v'_0 u''_0 v''_0$. Notice that a $\widehat{j}$-word appears in z_0 iff $j \in [-(h+1)..h+1]$.

We iteratively continue as above, defining words z_i, until reaching an iteration i, for which there is $k < i$, such that $p_i = p_k$.

We claim that the words $u = z_0 z_1 \cdots z_{k-1}$ and $v = z_k z_{k+1} \cdots z_{i-1}$ satisfy requirements i-iii w.r.t. q and $h+1$. The first two requirements are simply satisfied by the definition of u and v. As for the third requirement, we claim that when $\mathcal{R}^{p_k}$ runs on v, it visits disjoint set of states when reading v'_k and v''_k. This will provide the required 2^h different significant states, as $\mathcal{R}^{p_k}$ visits at least 2^{h-1} different significant states when reading each of v'_k and v''_k.

Indeed, assume, by way of contradiction, that $\mathcal{R}^{p_k}$ visits some state s both when reading v'_k and when reading v''_k. Let l' and r' be the parts of v'_k that $\mathcal{R}^{p_k}$ reads before and after reaching s, respectively, and l'' and r'' the analogous parts of v''_k. Now, define the infinite words $m' = u'_k\,(l'\,r')^\omega$, $m'' = u'_k\,l'\,(r''\,l'')^\omega$, and $m = u'_k\,(l'\,r''\,l''\,r')^\omega$.

Observe that since the language of $\mathcal{R}^{p_k}$ is the same as of $\mathcal{D}_n$, both m' and m'' are not accepted by $\mathcal{R}^{p_k}$, while m is accepted by $\mathcal{R}^{p_k}$. However, the set of states that are visited infinitely often in the run of $\mathcal{R}^{p_k}$ on m is the union of the sets of states that appear infinitely often in the runs of $\mathcal{R}^{p_k}$ on m' and m''. Hence, since the union of two Rabin-rejecting paths is Rabin-rejecting, the run of $\mathcal{R}^{p_k}$ on m should be rejecting, leading to a contradiction. ◀

Considering the translation of deterministic Muller to deterministic Streett automata, the family $\{\mathcal{D}_n\}_{n\geq 1}$ of DMWs of Figure 1 does not provide the required lower bound. Indeed, there is a DSW equivalent to $\mathcal{D}_n$ over the structure of $\mathcal{D}_n$ and having $2n$ Street acceptance pairs – for every $i \in [1..n]$, the pairs $\langle\{q_i\},\{q_{-i}\}\rangle$ and $\langle\{q_{-i}\},\{q_i\}\rangle$.

Yet, an exponential blowup can be shown by changing the acceptance condition of $\mathcal{D}_n$, such that the combination of two accepting cycles yields a rejecting cycle, as is done in the Muller automata $\mathcal{D}'_n$ of Figure 1.

▶ **Theorem 5.** *The translation of deterministic Muller automata to deterministic Streett automata involves a size blowup of at least $2^{\Omega(n)}$. In particular, there is a family $\{\mathcal{D}'_n\}_{n\geq 1}$ of DMWs with $2n+1$ states, $4n$ transitions, and $2n$ accepting sets, for which equivalent DSWs have at least 2^{n-1} states.*

Proof. The proof is completely analogous to the proof of Theorem 4, except for combining two accepting cycles rather than two rejecting ones; that is, the only change to the proof of Theorem 4 is in the last paragraph, which should be as follows.

Observe that since the language of $\mathcal{R}^{p_k}$ is the same as of $\mathcal{D}'_n$, both m' and m'' are accepted by $\mathcal{R}^{p_k}$, while m is not accepted by $\mathcal{R}^{p_k}$. However, the set of states that are visited infinitely often in the run of $\mathcal{R}^{p_k}$ on m is the union of the sets of states that appear infinitely often

in the runs of $\mathcal{R}^{p_k}$ on m' and m''. Hence, since the union of two Streett-accepting paths is Streett-rejecting, the run of $\mathcal{R}^{p_k}$ on m should be accepting, leading to a contradiction. ◀

3.3 From Muller To Nondeterministic Types

Our bounds for the translations of Muller automata to nondeterministic automata of the other types are the same when translating deterministic and when translating nondeterministic Muller automata. We show the upper bounds with respect to nondeterministic Muller automata, obviously holding also for deterministic automata, and the lower bounds with respect to deterministic Muller automata, obviously holding also for nondeterministic automata.

There is a well known translation of Muller automata to Büchi automata, involving a size blowup of $O(n^3)$, which can be improved to $O(n^2)$ when the target automaton is Streett.

We show a tight $\Omega(n^2)$ lower bound for the translation to Streett, and an $\Omega(n^2)$ lower bound for the translation to Rabin. The latter already holds for a Muller automaton with index 1. Combining the techniques of these two lower bounds, we show a tight $\Omega(n^3)$ lower bound for the translation to Büchi and parity automata.

3.3.1 Upper Bounds

The idea in translating a Muller automaton $\mathcal{A}$ with index k to a Büchi automaton $\mathcal{B}$ is to first "guess" which set S of states, out of the k possibilities, will be visited infinitely often. This contributes the 'first' n of the $O(n^3)$ construction. The second step is to "guess" when the states out of S are no longer visited. This step only doubles the automaton size. Then, having up to n states in S, $\mathcal{B}$ traverses n copies of the restriction of $\mathcal{A}$ to the states of S, for ensuring that all of the n states are visited infinitely often. This contributes the two other n's of the $O(n^3)$ construction.

▶ **Proposition 6.** *Muller automata of size n can be translated to Büchi automata of size in $O(n^3)$. In particular, for every NMW with l states, m transitions, and index k, there exists an equivalent NBW with kl^2 states and $k(l+1)m$ transitions.*

A translation to Streett automata is possible with only an $O(n^2)$ size blowup, using the Streett condition to enforce all of the relevant n states to be visited infinitely often.

▶ **Proposition 7.** *Muller automata of size n can be translated to Streett automata of size in $O(n^2)$. In particular, for every NMW with l states, m transitions, and index k, there exists an equivalent NSW with $2kl$ states, $3km$ transitions, and index kl.*

3.3.2 Lower Bounds

When we considered in Section 3.2 the translations to deterministic automata, we showed a lower bound on the number of states that a run r on some word w must visit, by adding up the already achieved lower bounds on sub-runs of r on subwords of w (Theorems 4 and 5). This technique cannot work when considering the translations to a nondeterministic automaton, as the automaton may have many runs on w, not necessarily containing the "best" runs on w's subwords.

For achieving a lower bound on the number of states in a nondeterministic Streett automaton $\mathcal{A}$, we define a new family $\{\mathcal{D}''_n\}$ of DMWs, as depicted in Figure 1, and concentrate on the accepting runs of $\mathcal{A}$. The DMW $\mathcal{D}''_n$ accepts words over the "alphabet" $[-n..n]$, in which the "letters" that appear infinitely often are exactly all of the "letters" between $-i$ and $n-i$, for some $i \in [1..n]$. Technically, a "letter" i is the finite word $a^i\#$ and $-i$ is $b^i\#$.

For getting the $\Omega(n^2)$ lower bound, we first show that every accepting run of $\mathcal{A}$ must visit infinitely often at least n different states. The reason is that $\mathcal{A}$ needs to count n consecutive a's or b's, as otherwise it will also accept illegal words with too many consecutive a's or b's. Next, we show that accepting runs on different words visit infinitely often disjoint sets of states. The reason stems from a property of the Streett condition, according to which the combination of two accepting cycles is accepting – if the runs had a common state, their combination would have accepted the combined word, which is rejected by $\mathcal{D}''_n$.

▶ **Theorem 8.** *The translation of deterministic Muller automata to nondeterministic Streett automata involves a size blowup of at least $\Omega(n^2)$. In particular, there is a family $\{\mathcal{D}''_n\}_{n\geq 1}$ of DMWs with $2n+1$ states, $4n$ transitions, and $n+1$ accepting sets, for which equivalent NSWs have at least $n^2/2$ states.*

Proof. Consider the family $\{\mathcal{D}''_n\}_{n\geq 1}$ of DMWs depicted in Figure 1, and let $\mathcal{A}$ be an NSW equivalent to $\mathcal{D}''_n$.

For every $i \in [0..n]$, define the word $w_i = (b^i \# a^{n-i} \#)^\omega$, which is accepted by $\mathcal{A}$, and let r_i be an accepting run of $\mathcal{A}$ on w_i. (For $i = 0$ and $i = n$, the first and last $\#$, respectively, are omitted from w_i's period.)

For showing that $\mathcal{A}$ has at least $n^2/2$ states, we will prove that I) for every $i \in [0..n]$, the run r_i visits infinitely often at least i different states, and II) for every $i \neq j \in [0..n]$, the states that r_i and r_j visit infinitely often are disjoint. (This will imply $\Sigma_{i=0}^{n} i = n(n+1)/2$ states.)

(I) Assume toward contradiction that for some $i \in [0..n]$, the run r_i visits less than i different states infinitely often. Then r_i makes a cycle c of length $m < i$ while reading only b's. Let q be a state that r_i visits infinitely often along the cycle c, and let p be the first position of w_i in which r_i visits q. Define the word w'_i that is derived from w_i by adding the finite word b^{nm} in position p. Observe that $\mathcal{A}$ accepts w'_i by a the run r' that starts like r_i, makes n times the cycle c in position p, and then continues like r_i. However, w'_i is not accepted by D''_n, leading to a contradiction.

(II) Assume toward contradiction that for some $i < j \in [0..n]$, both r_i and r_j visit some state s infinitely often. Let p and p' be positions of w_i in which r_i visits s, and between which r_i visits at least n times all the states that it will visit infinitely often. Let u be the subword of w_i between positions p and p'. Notice that w_i must contain both b^i and a^{n-i}.

Now, let w be the word that is derived from w_j by adding u in every position in which r_j visits s. Consider the run r of $\mathcal{A}$ on w that follows r_j, while making extra cycles from s back to itself in every position that u was added to w. Notice that the states that r visits infinitely often are the union of the states that r_i and r_j visit infinitely often. Hence, due to the property of the Street condition that the union of two accepting cycles is accepting, we have that r is accepting. Yet since w contains both b^j and a^{n-i} infinitely often, it is not accepted by $\mathcal{D}''_n$, leading to a contradiction. ◀

The proof of Theorem 8 is based on the fact that the union of two Streett accepting cycles is accepting. This does not hold for the Rabin condition, and therefore a different technique is needed for a lower bound in the translation to a nondeterministic Rabin automaton $\mathcal{A}$.

We should somehow take advantage of the dual property of the Rabin condition, according to which the union of two Rabin rejecting cycles is rejecting. Yet, there is no use in combining two rejecting runs, as $\mathcal{A}$ need not use their rejecting combination on a word that should be accepted, but rather use a different run that does accept it.

Our approach will be to construct a word w on which an accepting run r of $\mathcal{A}$ must visit at least n^2 different states, or else we can split r into rejecting runs whose union, which is

The deterministic Muller automata $\mathcal{M}_n$ and $\mathcal{M}'_n$

The states:

$\mathcal{M}_n$: Only the right side, namely $\{p_i, q_i \mid i \geq 0\}$

$\mathcal{M}'_n$: All states

The acceptance conditions:

$\mathcal{M}_n$: A single set with all its states, namely $\{p_i, q_i \mid i \geq 0\}$

$\mathcal{M}'_n$: The sets Q_i, for $i \in [0..n]$, where $Q_i = \{q_0\} \cup \{p_j, q_j \mid j \in [-i..n-i] \setminus \{0\}\}$

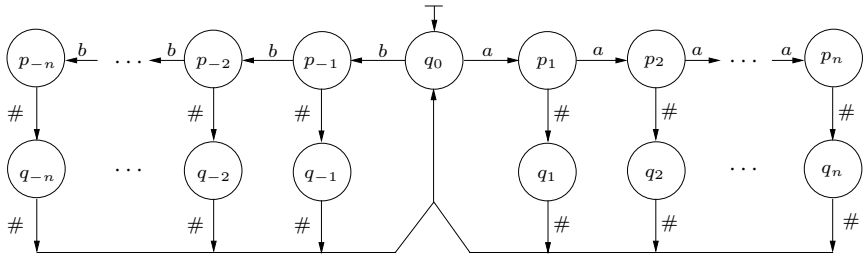


Figure 2 The Deterministic Muller automata $\mathcal{M}_n$ and $\mathcal{M}'_n$ of size in $O(n)$, for which equivalent nondeterministic Rabin and parity automata have, respectively, at least $n^2/2$ and $n^3/2$ states.

r, is also rejecting. For constructing such a word w, we need a different family of DMWs, in which a state q_i can be visited without visiting q_{i-1}. We define such a family of DMWs $\{\mathcal{M}_n\}$ in Figure 2.

▶ **Theorem 9.** *The translation of deterministic Muller automata, even with index 1, to nondeterministic Rabin automata involves a size blowup of at least $\Omega(n^2)$. In particular, there is a family $\{\mathcal{M}_n\}_{n\geq 1}$ of DMWs with $2n+1$ states, $3n$ transitions, and a single accepting set, for which equivalent NRWs have at least $n^2/2$ states.*

Proof. Consider the family $\{\mathcal{M}_n\}_{n\geq 1}$ of DMWs depicted in Figure 2, and let $\mathcal{A}$ be an NRW equivalent to $\mathcal{M}_n$.

Define the finite word $u = (a\#\#aa\#\# \cdots a^n\#\#)$, and the infinite word $w = u^\omega$. Notice that the length of u is bigger than $n^2/2$ and that $\mathcal{A}$ accepts w. We will show that an accepting run r of $\mathcal{A}$ on w must visit infinitely often at least $n^2/2$ different states, from which the required result immediately follows.

Assume toward contradiction that r visits the set S of states infinitely often, where $|S| < n^2/2$. Consider a simple cycle C of $\mathcal{A}$ along states in S. We claim that C is a rejecting cycle.

Indeed, consider a state q in C, let x be a finite word on which A can reach q, and let y be a finite word on which A^q can reach back q along C. The word y can either or not include the letter $\#$. If y does not include $\#$, then $\mathcal{M}_n$ does not accept the word yx^ω, since it has the infix a^{n+1}, on which M_n cannot run. If y does include $\#$, then $\mathcal{M}_n$ also does not accept the word yx^ω, since the subword a^n does not appear infinitely often in it. Hence, $\mathcal{M}_n$ does not accept yx^ω, implying that C is a rejecting cycle, as otherwise $\mathcal{A}$ would have accepted yx^ω.

Now, S is the union of its simple cycles, and since all of them are rejecting, by the property of the Rabin condition, so is S. Hence, the run r is rejecting. Contradiction. ◀

Parity automata are a special case of both Rabin and Streett automata. This suggests that we might be able to apply the lower bound techniques of both Theorem 8 and Theorem 9. Indeed, in Theorem 10 we show an $\Omega(n^3)$ lower bound for the translation of the deterministic

Muller automata $\{\mathcal{M}_n\}$ of Figure 2 to nondeterministic parity automata. We use the technique of Theorem 8 for showing that there are n different runs that visit disjoint states, and the technique of Theorem 9 for showing that each such run visits at least $n^2/2$ different states.

▶ **Theorem 10.** *The translation of deterministic Muller automata to nondeterministic parity automata involves a size blowup of at least $\Omega(n^3)$. In particular, there is a family $\{\mathcal{M}_n\}_{n\geq 1}$ of DMWs with $4n+1$ states, $6n$ transitions, and $n+1$ accepting sets, for which equivalent NRWs have at least $n^3/2$ states.*

Proof. Consider the family $\{\mathcal{M}_n\}_{n\geq 1}$ of DMWs depicted in Figure 2, and let $\mathcal{A}$ be an NPW equivalent to $\mathcal{M}_n$.

For every $i \in [0..n]$, define the finite word $u_i = (b\#\#bb\#\#\cdots b^i\#\#a\#\#aa\#\#\cdots a^{n-i}\#\#)$, and the infinite word $w = u_i^\omega$. (For $i = 0$ and $i = n$, the first and last $\#\#$, respectively, are omitted from u_i.) Notice that the length of u_i is bigger than $n^2/2$ and that $\mathcal{A}$ accepts w_i.

Analogously to the arguments in Theorem 9, for every $i \in [0..n]$, an accepting run r_i of $\mathcal{A}$ on w_i must visit infinitely often at least $n^2/2$ different states. Analogously to the arguments in Theorem 8, for every $i \neq j \in [0..n]$, accepting runs r_i and r_j of $\mathcal{A}$ on w_i and w_j, respectively, do not have any state in common. Hence, there are at least $n(n^2/2) = n^3/2$ states in $\mathcal{A}$. ◀

4 To Muller

We show that an exponential size blowup in the translations to nondeterministic Muller automata is inevitable, even when the source automaton is deterministic. Furthermore, translating nondeterministic automata to deterministic Muller automata, one cannot avoid the aforementioned exponential blowup, getting a doubly exponential size blowup.

The inevitable size blowups stem directly from the structure of the source automata, regardless of the alphabet and of the accepting condition – they already hold for looping automata, whole of whose states are accepting, over a fixed alphabet of three letters.

Exponential upper bounds are immediate: every (deterministic) automaton of the considered types has an equivalent (deterministic) Muller automaton over the same states and transitions. Thus, the Muller index can only be up to exponentially larger than the size of the input automaton. Considering the translations of nondeterministic types to deterministic Muller automata, the bound is tight for looping, weak, and co-Büchi automata, there is a gap between $2^{2^{\Omega(n)}}$ and $2^{2^{O(n \log n)}}$ for Büchi, and a gap between $2^{2^{\Omega(n)}}$ and $2^{2^{O(n^2 \log n^2)}}$ for parity, Rabin, and Streett.

▶ **Proposition 11.** *The translation of all (deterministic) automata to (deterministic) Muller automata involves a size blowup of up to $2^{O(n)}$. (All = Looping, weak, co-Büchi, Büchi, parity, Rabin, and Streett.)*

4.1 To Nondeterministic Muller

An exponential size blowup in the translation of deterministic Büchi automata to nondeterministic Muller automata is shown in [17]. They provide[1] a family $\{L_n\}$ of languages

[1] In [17], there is a statement of the lower bound without providing the explicit languages and Büchi automata. They refer to a lemma about the complementation of Muller automata, and claim that

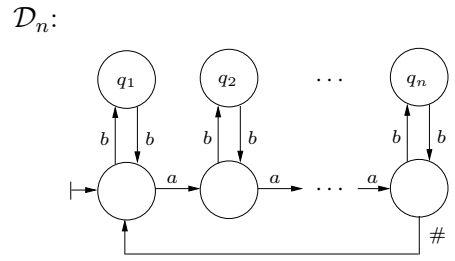


Figure 3 Deterministic looping automata of size in $O(n)$, for which equivalent nondeterministic Muller automata need an index of at least 2^n.

over alphabets of length $2n$, such that L_n is recognized by a Büchi automaton with $n+2$ states and n^2+n+2 transitions, while equivalent Muller automata have an index of at least 2^n.

We improve the result by providing a different family $\{L'_n\}$ of languages over a fixed alphabet, such that L'_n is recognized by a looping automaton with $2n$ states and $3n$ transitions, as depicted in Figure 3, while equivalent Muller automata need an index of at least 2^n.

▶ **Theorem 12.** *The translation of deterministic looping automata to nondeterministic Muller automata involves a size blowup of at least* $2^{\Omega(n)}$.

Proof. For every positive $n \in \mathbb{N}$, consider the DLW $\mathcal{D}_n$ of Figure 3, which we dub $\mathcal{D}$, and let $\mathcal{M}$ be an NMW equivalent to $\mathcal{D}$. We denote by L_a the language of words with infinitely many a's.

We first classify the states of $\mathcal{M}$ according to their connection with the q_i states of $\mathcal{D}$. Formally, for every $i \in [1..n]$, we define the set S_i of states of $\mathcal{M}$ to include exactly the states s for which $L(\mathcal{M}^s) \cap L(\mathcal{D}^{q_i}) \cap L_a \neq \emptyset$.

Observe that for every $i \in [1..n]$ and finite word u, if $\mathcal{M}$ visits a state $s \in S_i$ after reading u along an accepting run of $\mathcal{M}$ on a word with infinitely many a's, then $\mathcal{D}(u) = q_i$. This follows from the structure of $\mathcal{D}$, according to which there is no word $v \in L_a$, such that $v \in L(\mathcal{D}^{q_i}) \cap L(\mathcal{D}^p)$ for $p \neq q_i$. This also implies that the sets $S_1, \ldots, S_n$ are mutually disjoint.

We continue with showing that $\mathcal{M}$'s index must be at least 2^n. For every subset $H \subseteq [1..n]$, there is a word $w_H \in L(\mathcal{D}) \cap L_a$, such that for every $i \in [1..n]$, $\mathcal{D}$'s run on w_H visits q_i infinitely often iff $q_i \in H$. Now, for every $i \in H$, there must be a state $s \in S_i$ that is visited infinitely often by an accepting run r_H of $\mathcal{M}$ on w_H, as infinitely often $(\mathcal{D}^{q_i}) \cap L_a \neq \emptyset$. Let F_H be an accepting set of $\mathcal{M}$ according to which r_H is accepted.

Assume, by way of contradiction, that there are two subsets $H \neq H'$ such that $F_H = F_{H'}$. Without loss of generality, consider a number $i \in H' \setminus H$. Then, since there is a state $s \in S_i \cap F_{H'}$ and $F_H = F_{H'}$, it follows that s is visited infinitely often in the run r_H of $\mathcal{M}$ on w_H. Thus, $\mathcal{D}$ visits q_i infinitely often in its run over w_H, contradicting the definition of w_H, as $i \notin H$. ◀

analogous languages provide the lower bound. The languages described here in the name of [17] are those assumed to be the analogous ones.

4.2 To Deterministic Muller

Following Theorem 12, the translation of deterministic automata, of all relevant types, to deterministic Muller automata is in $2^{\Theta(n)}$.

We shall look into the translations of nondeterministic automata to deterministic Muller automata. These translations might involve a doubly exponential size blowup, as the determinization process exponentially enlarges the number of states, and the Muller index might be exponential in the latter number of states. This is indeed the case, and the double exponent stems directly from the automaton structure. It already holds for a looping automaton (all of whose states are accepting) over a fixed alphabet.

4.2.1 Upper Bounds

A co-Büchi automaton of size n can be translated to a deterministic Muller automaton with $2^{O(n)}$ states [12], on top of which the index is in $2^{2^{O(n)}}$.

▶ **Proposition 13.** *Looping, weak, and co-Büchi automata of size n can be translated to deterministic Muller automata of size in $2^{2^{O(n)}}$.*

A Büchi automaton of size n can be translated to a deterministic Muller automaton with $2^{O(n \log n)}$ states [17], on top of which the index is in $2^{2^{O(n \log n)}}$.

▶ **Proposition 14.** *Büchi automata of size n can be translated to deterministic Muller automata of size in $2^{2^{O(n \log n)}}$.*

Parity and Rabin automata of size n can be translated to a Büchi automaton of size in $O(n^2)$, and then determinized into a Muller automaton. For Streett automata, the above procedure does not work as there is an exponential blowup in the translation of an NSW to an NBW. Yet, one may determinize the Streett automaton directly into a deterministic Muller automaton with $2^{O(n^2 \log n^2)}$ states [18, 15].

▶ **Proposition 15.** *Parity, Rabin, and Streett automata of size n can be translated to deterministic Muller automata of size in $2^{2^{O(n^2 \log n^2)}}$.*

4.2.2 Lower Bounds

A doubly exponential lower bound is already shown in [17], however it uses a family of Büchi automata over alphabets of length exponential in the number of states. Hence, it is reasonable to view the size of the resulting deterministic Muller automaton as only singly exponential in the size of the original automaton. Safra's languages [17]: For every $n > 0$, let $\mathcal{S}_n = 2^{[1..n]}$ and $\Sigma_n = \mathcal{S}_n \cup [1..n]$, and define the language

$$L_n = \{Y_0 y_0 Y_1 y_1 \ldots \mid \text{for all } i,\ Y_i \in \mathcal{S}_n \text{ and } y_i \in Y_i\}.$$

One can change Safra's languages to use an alphabet of length linear in the number of states, by replacing the subset-letters Y_i with finite strings, separated with a dedicated symbol #. For avoiding the need to count the finite-strings length, these strings will be of an arbitrary length, "encoding" more than the original exponential alphabet. Formally, for every $n > 0$, let $\Sigma_n = \{\#, 1, 2, \ldots, n\}$, and define the language

$$L'_n = \{Y_0 \# y_0 \# Y_1 \# y_1 \# \ldots \mid \text{for all } i,\ Y_i \in [1..n]^* \text{ and } y_i \in [1..n] \text{ appears in } Y_i\}.$$

For achieving a $2^{2^{\Omega(n)}}$ lower bound, we further change the above languages, using a fixed alphabet. Except for encoding the linear alphabet by a fixed one, analogously to

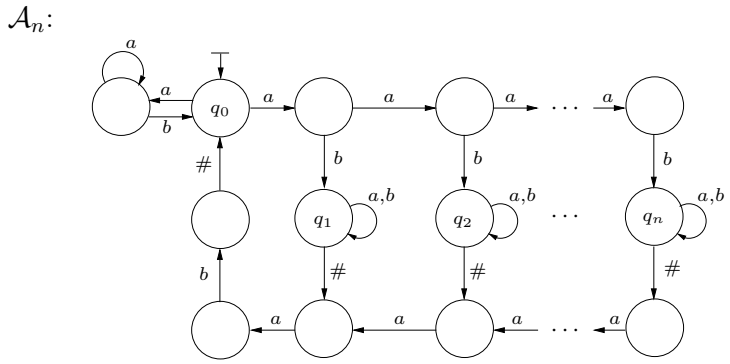


Figure 4 Looping automata of size in $O(n)$, for which equivalent deterministic Muller automata need an index of at least 2^{2^n}.

the way Löding encoded Michel's language for the Büchi complementation lower bound [9], we simplify the languages, in order to be recognized by looping automata, as depicted in Figure 4.

▶ **Theorem 16.** *The translation of nondeterministic looping automata to deterministic Muller automata involves a size blowup of at least* $2^{2^{\Omega(n)}}$.

Proof. For every positive $n \in \mathbb{N}$, consider the NLW $\mathcal{A}_n$ of Figure 4, and let $\mathcal{M}$ be a DMW equivalent to $\mathcal{A}_n$.

In the scope of this proof, we say that the "encoding" of a number $i \in \mathbb{N}$, denoted by $\widehat{i}$, is the finite string $a^i b$. For example, $\widehat{2} = aab$ and $\widehat{0} = b$.

Consider the set S of states of $\mathcal{M}$ that are reachable after reading a finite word u, such that: i) u ends with #, ii) there is an odd number of #'s in u, and iii) there exists an infinite word v, such that $u \cdot v \in L(\mathcal{A}_n)$. Then S has at least 2^n states, corresponding to the subsets of numbers in $[1..n]$ whose encodings appear in the suffix of u from the last even occurrence of a # onwards, as shown below.

Indeed, assume toward contradiction a state q of $\mathcal{M}$ that is reachable after reading two finite words, u_1 and u_2, satisfying constraints i-iii above, and whose suffixes include encodings of different numbers in $[1..n]$. Without loss of generality, there is a number $i \in [1..n]$ that is encoded in the suffix of u_1 and not in the suffix of u_2. Let $v = (\widehat{i} \cdot \#)^\omega$. Since the word $w_1 = u_1 \cdot v \in L(\mathcal{M})$, there is an accepting run of $\mathcal{M}^q$ on v. Thus, $\mathcal{M}$ also accepts the word $w_2 = u_2 \cdot v$, which is not in $L(\mathcal{M})$, leading to contradiction.

We continue with showing that $\mathcal{M}$'s index must be at least 2^{2^n-1}. Consider the set $H = \{\emptyset, H_1, H_2, \ldots, H_{2^n-1}\}$ of subsets of $[1..n]$. For every $i \in [1..2^n - 1]$, let h_i be the minimal number in H_i, and let $\widehat{H_i}$ be some finite word that is the concatenation of all the words $\widehat{j}$, such that $j \in H_i$. As shown above, for every element H_i of $H \setminus \{\emptyset\}$, there is a corresponding state s_{H_i} in S. We will show that $\mathcal{M}$ must have an acceptance set for every subset of S.

For every subset $Z = \{H_{i_1}, H_{i_2}, \ldots, H_{i_{|z|}}\}$ of $H \setminus \{\emptyset\}$, consider the infinite word $w_Z = (\widehat{H_{i_1}}\#\widehat{h_{i_1}}\#\widehat{H_{i_2}}\#\widehat{h_{i_2}} \cdots \widehat{H_{i_{|z|}}}\#\widehat{h_{i_{|z|}}}\#)^\omega$. By the structure of $\mathcal{A}_n$, $w_Z \in L(\mathcal{A}_n) = L(\mathcal{M})$. By the definition of the states in S, the accepting run of $\mathcal{M}$ on w_Z visits a state s_x of S infinitely often if and only if $x \in Z$. Hence, $\mathcal{M}$ has a different acceptance set for every subset of $H \setminus \{\emptyset\}$, implying an index of least 2^{2^n-1}. ◀

References

1. D. Angluin, U. Boker, and D. Fisman. Families of DFAs as acceptors of omega-regular languages. In *Proc. of MFCS*, pages 11:1–11:14, 2016.
2. U. Boker and O. Kupferman. Translating to co-Büchi made tight, unified, and useful. *ACM Trans. Comput. Log.*, 13(4):29:1–29:26, 2012.
3. U. Boker, O. Kupferman, and A. Rosenberg. Alternation removal in Büchi automata. In *Proc. of ICALP*, volume 6199, pages 76–87, 2010.
4. J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.
5. T. Colcombet and K. Zdanowski. A tight lower bound for determinization of transition labeled Büchi automata. In *Proc. of ICALP*, pages 151–162, 2009.
6. Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 60–65. ACM Press, 1982.
7. O. Kupferman and N. Piterman. Lower bounds on witnesses for nonemptiness of universal co-Büchi automata. In *Proc. of FoSSaCS*, volume 5504 of *LNCS*, pages 182–196. Springer, 2009.
8. O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(2):408–429, 2001.
9. C. Löding. Optimal bounds for the transformation of omega-automata. In *Proc. of FSTTCS*, volume 1738 of *LNCS*, pages 97–109, 1999.
10. R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.
11. M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.
12. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
13. A. W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, volume 208 of *LNCS*, pages 157–168. Springer, 1984.
14. D. E. Muller. Infinite sequences and finite machines. In *Proc. 4th IEEE Symp. on Switching Circuit Theory and Logical design*, pages 3–16, 1963.
15. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3):5, 2007.
16. M. O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
17. S. Safra. *Complexity of automata on infinite objects.* PhD thesis, Weizmann Institute of Science, 1989.
18. S. Safra. Exponential determinization for ω-automata with strong-fairness acceptance condition. In *Proc. 24th ACM Symp. on Theory of Computing*, 1992.
19. S. Schewe. Büchi complementation made tight. In *Proc. 26th Symp. on Theoretical Aspects of Computer Science*, volume 3 of *LIPIcs*, pages 661–672. Schloss Dagstuhl, 2009.
20. S. Schewe and T. Varghese. Determinising parity automata. In *Proc. of MFCS*, pages 486–498, 2014.
21. R. S. Streett. Propositional dynamic logic of looping and converse. *Information and Control*, 54:121–141, 1982.
22. Q. Yan. Lower bounds for complementation of omega-automata via the full automata technique. *Logical Methods in Computer Science*, 4(1), 2008.

Stone Duality and the Substitution Principle*

Célia Borlido[1], Silke Czarnetzki[2], Mai Gehrke[3], and Andreas Krebs[4]

1 **IRIF, CNRS and Université Paris Diderot, Paris, France**
2 **Wilhelm-Schickard Institut, Universität Tübingen, Tübingen, Germany**
3 **IRIF, CNRS and Université Paris Diderot, Paris, France**
4 **Wilhelm-Schickard Institut, Universität Tübingen, Tübingen, Germany**

Abstract

In this paper we relate two generalisations of the finite monoid recognisers of automata theory for the study of circuit complexity classes: Boolean spaces with internal monoids and typed monoids. Using the setting of stamps, this allows us to generalise a number of results from algebraic automata theory as it relates to Büchi's logic on words. We obtain an Eilenberg theorem, a substitution principle based on Stone duality, a block product principle for typed stamps and, as our main result, a topological semidirect product construction, which corresponds to the application of a general form of quantification. These results provide tools for the study of language classes given by logic fragments such as the Boolean circuit complexity classes.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases $\mathcal{C}$-variety of languages, typed monoid, Boolean space with an internal monoid, substitution principle, semidirect product.

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.13

1 Introduction

Complexity theory and the theory of regular languages are intimately connected through logic. As with classes of regular languages, many computational complexity classes are model classes of appropriate logic fragments on finite words [12]. For example, $\mathrm{AC}^0 = FO[\mathrm{arb}]$, $\mathrm{ACC}^0 = (FO + MOD)[\mathrm{arb}]$, and $\mathrm{TC}^0 = MAJ[\mathrm{arb}]$ where arb is the set of all predicates on the positions of a word, FO is first-order logic, and MOD and MAJ stand for the *modular* and *majority* quantifiers, respectively. On the one hand, the presence of *arbitrary (numerical) predicates*, and on the other hand, the presence of the *majority quantifier* is what brings one far beyond the scope of the profinite algebraic theory of regular languages.

Most results in complexity theory are proved with combinatorial, probabilistic, and algorithmic methods [18]. However, there are a few connections with the topo-algebraic tools for regular languages. A famous result of Barrington, Compton, Straubing, and Thérien [2] states that a regular language is in AC^0 if and only if its syntactic homomorphism is quasi-aperiodic. Although this result relies on [5] and no purely algebraic proof is known, being able to characterise the class of regular languages in AC^0 gives some hope that the non-uniform classes might be amenable to treatment by the generalised topo-algebraic methods.

Indeed, the hope is that one can generalise the tools of algebraic automata theory. In the paper *Logic Meets Algebra: the case of regular languages* [17], Thérien and Tesson lay

* This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.670624) and from the DFG Emmy Noether program (KR 4042/2).

out the theory used to characterise logic classes in the setting of regular languages in terms of their recognisers. Here we add topology to the picture, using Stone duality, to obtain corresponding tools that apply beyond the setting of regular languages.

Two ways of breaking the regular barrier already exist. Typed monoids [14], which still have a finite component, must be studied in families, while Boolean spaces with internal monoids [7, 10] generalise the profinite monoids of the classical theory and provide single topological objects of study relative to each alphabet. In Section 3 we relate these via an Eilenberg theorem improving on [3] by identifying tighter closure properties.

Using typed stamps, i.e. morphisms from finitely generated free monoids to typed monoids, we show that the Stone dual of predicate substitution is given by a transduction (Section 4), and that transduced languages are the ones recognised by a generalised version of the block product (Section 5). Thus we identify this important connection between logical generation and algebraic recognition as a case of Stone duality in a very general setting.

The topological recognisers provide access to equations as in Eilenberg-Reiterman theory [9], and thus to tools for separation and, in the finite case, decidability. The main result of the paper, Theorem 23, is a general form of the classical result by Almeida and Weil [1], which provides a semidirect product construction for Boolean spaces with dense monoids characterising the block product of varieties of typed stamps. Finally, in Section 7, we illustrate how these tools may be applied in the study of Boolean circuit classes.

2 Preliminaries on Stone duality

In this section we present the basics of Stone duality as used in the rest of the paper. See [6, 8] for an adapted introduction or [13] for further details.

The most basic duality we use, also known as *discrete duality*, provides a correspondence between powerset Boolean algebras (these are the complete and atomic Boolean algebras) and sets. Given such a Boolean algebra $\mathcal{B}$, its dual is its set of atoms, denoted $\mathsf{At}(\mathcal{B})$ and, given a set X, its dual is the Boolean algebra $\mathcal{P}(X)$. Clearly going back and forth yields isomorphic objects. If $h\colon \mathcal{B} \to \mathcal{A}$ preserves arbitrary meets and joins, the dual of h, denoted $\mathsf{At}(h)\colon \mathsf{At}(\mathcal{A}) \to \mathsf{At}(\mathcal{B})$ is given by the adjunction:

$$\forall\, a \in \mathcal{A} \;\text{ and }\; \forall\, x \in \mathsf{At}(\mathcal{B}) \quad \big(\, \mathsf{At}(h)(x) \leq a \iff x \leq h(a) \,\big)\,.$$

For example, if $\iota\colon \mathcal{B} \hookrightarrow \mathcal{P}(X)$ is the inclusion of a finite Boolean subalgebra of a powerset, then $\mathsf{At}(\iota)\colon X \twoheadrightarrow \mathsf{At}(\mathcal{B})$ is the quotient map corresponding to the finite partition of X given by the atoms of $\mathcal{B}$. Conversely, given a function $f\colon X \to Y$, the dual is just $\mathcal{P}(f) = f^{-1}\colon \mathcal{P}(Y) \to \mathcal{P}(X)$.

Generally Boolean algebras do not have enough atoms, and we have to consider *ultrafilters* instead (which may be seen as 'searches downwards' for atoms). Given an arbitrary Boolean algebra $\mathcal{B}$, an ultrafilter of $\mathcal{B}$ is a non-empty subset μ of $\mathcal{B}$ satisfying:

- μ is an upset, i.e., $a \in \mu$ and $a \leq b$ implies $b \in \mu$;
- μ is closed under finite meets, i.e., $a, b \in \mu$ implies $a \wedge b \in \mu$;
- for all $a \in \mathcal{B}$ exactly one of a and $\neg a$ is in μ.

Here, we will denote the set of ultrafilters of $\mathcal{B}$ by $X_{\mathcal{B}}$, and we will consider it as a topological space equipped with the topology generated by the sets $\widehat{a} = \{\mu \in X_{\mathcal{B}} \mid a \in \mu\}$ for $a \in \mathcal{B}$. The last property in the definition of ultrafilters implies that these basic open sets are also closed (and thus *clopen*). The resulting spaces are compact, Hausdorff, and have a basis of clopens. Such spaces are called *Boolean spaces*. Conversely, given a Boolean space, its clopen subsets form a Boolean algebra and one can show that going back and forth

results in isomorphic objects. Given a homomorphism $h\colon \mathcal{A} \to \mathcal{B}$ between Boolean algebras, one can show that the inverse image of an ultrafilter is an ultrafilter and thus h^{-1} induces a continuous map $X_{\mathcal{B}} \to X_{\mathcal{A}}$. Finally, given a continuous function $f\colon X \to Y$, the inverse map restricts to clopens and yields a homomorphism of Boolean algebras.

Boolean spaces are also the *profinite sets*, see Appendix A. This is fundamental for the toggling between the two generalisations of finite monoids given below. Finally, for a set S, the dual space of $\mathcal{P}(S)$, denoted $\beta(S)$, is the *Stone-Čech compactification* of S. For each $s \in S$, the set $\iota(s) = \{P \in \mathcal{P}(S) \mid s \in P\}$ is the *principal ultrafilter generated by* s, and the induced map $\iota : S \hookrightarrow \beta(S)$ is an injection with dense image.

3 Recognition of languages

The notion of recognition originates in automata theory where the classical recognisers are monoid morphisms into finite monoids. Many important classes of regular languages correspond to *pseudovarieties of finite monoids*, that is, classes closed under homomorphic images, subalgebras, and finite products. Eilenberg's theorem characterises the corresponding classes of regular languages, called *varieties of regular languages*. All the languages recognised by a monoid of a given pseudovariety, no matter the recognising morphism used, belong to the corresponding variety of languages. However, not all language classes of interest form varieties. In fact, many classes corresponding to fragments of logic contain all the languages recognised by one morphism into a finite monoid but not the ones recognised via another morphism into the same monoid. It follows that such classes are not closed under inverse images for arbitrary morphisms between finitely generated free monoids, and one has to keep track of classes of morphisms from free monoids to finite ones, so-called *stamps*. Stamps, and a notion of pseudovariety of stamps, were introduced by Straubing [16] to study language classes coming from logic on words.

If one is interested in classes that contain non-regular languages, one needs more complex recognisers based on infinite monoids. In [7] a notion of compact topological recognisers was introduced and in [10] a more duality-friendly variant, so-called Boolean spaces with internal monoids, was given. These recognisers, based on Boolean spaces, provide a notion of recognition appropriate also for non-regular languages, but the finiteness is lost. Independently [3] introduced typed monoids, which are infinite monoids equipped with a finite set-quotient. These have a finite component but must be studied in families. Here we show how these two notions fit together and provide a variant of Eilenberg's variety theorem in the setting of stamps for languages that are not necessarily regular.

3.1 Generalising finite monoids

The basic recognisers of automata theory are finite monoids. Here, we consider recognisers which separate the attributes of being finite and of being a monoid.

▶ **Definition 1.** A *typed monoid* is a tuple $\mathsf{R} = (M, p, X)$, where X is a finite set, M is a monoid and $p\colon M \twoheadrightarrow X$ is a surjective set function. A *typed submonoid* of R is given by a submonoid N of M by restriction of p and its image, $\mathsf{R}|_N = (N, p|_N, p[N])$. A *morphism of typed monoids*, $\Phi\colon (M, p, X) \to (N, q, Y)$, is a pair $\Phi = (g, \varphi)$ where $g\colon M \to N$ is a monoid morphism and $\varphi\colon X \to Y$ is a set function, so that $\varphi \circ p = q \circ g$. The *image* of Φ is the typed submonoid of (N, q, Y) given by $g[N]$. We say that (M, p, X) *recognises* the language $L \subseteq A^*$ when there is a monoid morphism $\mu : A^* \to M$ and $C \subseteq X$ such that $L = (p \circ \mu)^{-1}(C)$.

For instance, $L_{\mathrm{Eq}} = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$, where $|w|_a$ and $|w|_b$ stand, respectively, for the number of a's and of b's in w, is recognised by $(\mathbb{Z}, p, \{0,1\})$, where $p(n) = 1$ if and only if $n = 0$. The recognising morphism h_{Eq} sends a to 1 and b to -1 and $L_{\mathrm{Eq}} = h_{\mathrm{Eq}}^{-1}(\{1\})$.

Finite monoids provide invariants that are useful in describing and understanding classes of regular languages. Their pertinence arises from the fact that many classes of languages of interest are closed under *quotients* by words. That is, if $L \subseteq A^*$ is in the class and $u \in A^*$ then $u^{-1}L = \{w \in A^* \mid uw \in L\}$ and $Lu^{-1} = \{w \in A^* \mid wu \in L\}$ are also in the class. Note that the Boolean algebra of all languages recognised by a morphism $\mu\colon A^* \to M$ is closed under quotients. However, this is not true for typed monoids in general and, given a non-regular language, the closure under quotienting is necessarily infinite, so we need to capture infinite Boolean subalgebras of $\mathcal{P}(M)$ for M an infinite monoid. For this purpose, we recall that a *biaction* of the monoid M on a set X is given by a two-sided action map $\alpha\colon M \times X \times M \to M$ satisfying $\alpha(1,x,1) = x$ where 1 is the identity element of M and $\alpha(m_1, \alpha(m_1', x, m_2'), m_2) = \alpha(m_1 m_1', x, m_2' m_2)$ for all $m_1, m_1', m_2', m_2 \in M$. The *left component* of the action at m is the map $\lambda_m\colon X \to X$ given by $x \mapsto \alpha(m,x,1)$, while the *right component* at m is $\rho_m\colon x \mapsto \alpha(1,x,m)$. As derived in [10], the following topological notion captures Boolean subalgebras of a monoid which are closed under the quotient operations.

▶ **Definition 2.** A *Boolean space with an internal monoid* (BiM) is a triple (M,p,X) where X is a Boolean space equipped with a biaction of a monoid M whose right and left components at each $m \in M$ are continuous and a map $p\colon M \to X$ which has dense image and is a morphism of sets with M-biactions. That is, for each $m \in M$, the following diagrams commute:

$$\begin{array}{ccc} M & \xrightarrow{p} & X \\ {\scriptstyle \ell_m}\downarrow & & \downarrow{\scriptstyle \lambda_m} \\ M & \xrightarrow{p} & X \end{array} \qquad \begin{array}{ccc} M & \xrightarrow{p} & X \\ {\scriptstyle r_m}\downarrow & & \downarrow{\scriptstyle \rho_m} \\ M & \xrightarrow{p} & X \end{array}$$

with $\ell_m\colon n \mapsto m \cdot n$ and $r_m\colon n \mapsto n \cdot m$ the components at m of the biaction of M on itself.

▶ **Example 3.** The Boolean algebra closed under quotients that is generated by L_{Eq} defined above is recognised by the BiM $(\mathbb{Z}, p, \mathbb{Z} \cup \{\infty\})$ via the morphism h_{Eq}. Here, $\mathbb{Z} \cup \{\infty\}$ is the one-point compactification of $\mathbb{Z}$, p is just the inclusion map, and the left and right actions are given by the usual addition on $\mathbb{Z}$ augmented by $n + \infty = \infty = \infty + n$, for every $n \in \mathbb{Z}$.

The weaker structure (M,p,X) where M is a monoid, X is a Boolean space, and p has dense image we will call a *Boolean space with a dense monoid*, or simply a *B-monoid*. Note that typed monoids are precisely the finite B-monoids (those for which X is finite). Moreover, a *B-submonoid* of (M,p,X) is given by a submonoid N of M by restricting p and viewing it as a map into the topological closure of its image.

Alternatively, we can capture infinite Boolean subalgebras of $\mathcal{P}(M)$ closed under the quotients using subfamilies $\mathcal{S} \subseteq \mathcal{T}_M = \{(M,p,X) \mid (M,p,X) \text{ is a typed monoid}\}$. For this purpose, we define a quasiorder on $\mathcal{T}_M$ given by $(M,p,X) \geq (M,q,Y)$ provided (M,q,Y) is a quotient of (M,p,X) for which the map g is the identity. The family $\mathcal{S}$ is then a *downset* of $\mathcal{T}_M$ provided it is closed under such quotients, and $\mathcal{S}$ is *directed* provided $(M,p,X), (M,q,Y) \in \mathcal{S}$ implies the existence of $(M,r,Z) \in \mathcal{S}$ with quotient maps $p'\colon Z \to X$ and $q'\colon Z \to Y$ so that $p = p' \circ r$ and $q = q' \circ r$. Also, we say that $\mathcal{S}$ is *multiplicative* provided for all $(M,p,X) \in \mathcal{S}$ and $m \in M$, there are $(M,q,Y), (M,r,Z) \in \mathcal{S}$ and maps $\lambda_m\colon Y \to X$ and $\rho_m\colon Z \to X$ so that $\lambda_m \circ q(m') = p(mm')$ and $\rho_m \circ r(m') = p(m'm)$, for every $m' \in M$.

Stone duality yields the following proposition.

▶ **Proposition 4.** *For a monoid M, there are bijections between each of the following:*
1. *the set of Boolean subalgebras closed under quotients $\mathcal{B} \subseteq \mathcal{P}(M)$;*
2. *the set of directed and multiplicative downsets $\mathcal{S} \subseteq \mathcal{T}_M$ of typed monoids based on M;*
3. *the set of Boolean spaces with internal monoids (M, p, X) based on M.*

Proof Sketch. The bijections are given as follows. Given a Boolean subalgebra $\mathcal{B}$ of $\mathcal{P}(M)$ closed under quotients, the corresponding directed and multiplicative downset is

$$\mathcal{S}_{\mathcal{B}} = \{(M, \mathsf{At}(\iota), \mathsf{At}(\mathcal{B}')) \mid \mathcal{B}' \subseteq \mathcal{B} \text{ is a finite Boolean subalgebra and } \iota\colon \mathcal{B}' \hookrightarrow \mathcal{P}(M)\}.$$

In turn, given a directed and multiplicative downset of typed monoids $\mathcal{S} \subseteq \mathcal{T}_M$, the limit of the projective system of maps $p_i : M \twoheadrightarrow X_i$ with $(M, p_i, X_i) \in \mathcal{S}$ defines a BiM, see Appendix A. Finally, given a BiM (M, p, X), the corresponding Boolean algebra is

$$\mathcal{B}_{(M,p,X)} = \{p^{-1}(K) \mid K \subseteq X \text{ is clopen}\}. \qquad \blacktriangleleft$$

3.2 Stamps for non-regular languages

▶ **Definition 5.** A *Boolean space with a dense stamp* (or *B-stamp* for short) is a tuple $\mathsf{R} = (A, \mu, M, p, X)$ where $\mu : A^* \twoheadrightarrow M$ is a monoid quotient and (M, p, X) is a B-monoid. A B-stamp is called a BiM *presentation* when (M, p, X) is a BiM and a *typed stamp* when (M, p, X) is a typed monoid.

A morphism between B-stamps $\mathsf{R} = (A, \mu, M, p, X)$ and $\mathsf{S} = (B, \nu, N, q, Y)$ is a triple $\Phi = (h, g, \varphi)$, where $h : A^* \to B^*$ and $g : M \to N$ are monoid morphisms, $\varphi : X \to Y$ is a continuous function, and the following diagram commutes:

$$\begin{array}{ccccc} A^* & \xrightarrow{\mu} & M & \xrightarrow{p} & X \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle \varphi} \\ B^* & \xrightarrow{\nu} & N & \xrightarrow{q} & Y \end{array}$$

When h is the identity on A^*, we say that S *factors through* R. Further, each morphism $h : A^* \to B^*$ defines a *substamp* $(h[A], \nu', N', q', Y')$ of S, where the elements of $h[A]$ are regarded as letters, $\nu' : h[A]^* \to N'$ is the surjective co-restriction of the monoid morphism sending $u \in h[A]$ to $\nu(u)$, and (N', q', Y') is the sub of (N, q, Y) given by N'.

B-stamps R encode two types of behaviour: algebraic behaviour given by the *monoid presentation* (A, μ, M) and topological behaviour given by the B-monoid (M, p, X). The interplay between these two is a key ingredient in this paper. We say that a language L is *recognised* by R provided there exists a clopen subset $C \subseteq X$ such that $L = (p \circ \mu)^{-1}(C)$. Notice that the set of all languages recognised by a B-stamp always forms a Boolean algebra.

▶ **Definition 6.** Let $\mathcal{B} \subseteq \mathcal{P}(A^*)$ be a Boolean algebra of languages. Then the *syntactic B-stamp* of $\mathcal{B}$ is $\mathsf{R}_{\mathcal{B}} = (A, \mu_{\mathcal{B}}, M_{\mathcal{B}}, p_{\mathcal{B}}, X_{\mathcal{B}})$ where $X_{\mathcal{B}}$ is the dual space of $\mathcal{B}$ and $M_{\mathcal{B}}{=}A^*/{\sim_{\mathcal{B}}}$ where the *syntactic congruence* $\sim_{\mathcal{B}}$ is given by

$$w \sim_{\mathcal{B}} w' \iff \forall L \in \mathcal{B},\ \forall u, v \in A^* \quad (uwv \in L \iff uw'v \in L).$$

The quotient map $\mu_{\mathcal{B}}\colon A^* \twoheadrightarrow A^*/{\sim_{\mathcal{B}}}$ is the *syntactic morphism* of $\mathcal{B}$. The restriction to A^* of the dual of the inclusion $\iota\colon \mathcal{B} \hookrightarrow \mathcal{P}(A^*)$ is a map $\tilde{\iota} : A^* \to X_{\mathcal{B}}$ with dense image. Since $\sim_{\mathcal{B}}$ is the least monoid congruence containing the kernel of $\tilde{\iota}$, it factors through $\mu_{\mathcal{B}}$. That is, there is a map $p_{\mathcal{B}}\colon M_{\mathcal{B}} \to X_B$ so that $\tilde{\iota} = p_{\mathcal{B}} \circ \mu_{\mathcal{B}}$. Clearly, the image of $p_{\mathcal{B}}$ is dense in X.

▶ **Proposition 7.** *A B-stamp R recognises a Boolean algebra closed under quotients $\mathcal{B}$ if and only if the syntactic B-stamp of $\mathcal{B}$ factors through R.*

3.3 $\mathcal{C}$-varieties of languages and an Eilenberg theorem

As mentioned earlier, we are interested in classes of languages that may not be closed under preimages of arbitrary morphisms. In what follows, we fix a class $\mathcal{C}$ of morphisms between finitely generated free monoids which is closed under composition and contains all length-preserving morphisms (i.e. the ones sending generators to generators), written *lp-morphism* (we will see why *lp*-morphisms are important in the treatment of logic on words in Section 4.2). A *$\mathcal{C}$-variety of languages* is an assignment, for each finite alphabet A, of a Boolean algebra closed under quotients $\mathcal{V}(A)$ of languages over A such that, for every morphism $h : B^* \to A^*$ in $\mathcal{C}$, if $L \in \mathcal{V}(A)$ then $h^{-1}(L) \in \mathcal{V}(B)$. In this subsection, we identify the classes of typed stamps that correspond to $\mathcal{C}$-varieties of languages. For this, we need some definitions.

A morphism $\Phi = (h, g, \varphi)$ of typed stamps is a *$\mathcal{C}$-morphism* provided $h \in \mathcal{C}$ and a *$\mathcal{C}$-substamp* is a typed substamp given by $h \in \mathcal{C}$. Let $\{\mathsf{R}_i = (A, \mu_i, M_i, p_i, X_i)\}_{i\in I}$ be a family of B-stamps. Their *product* is the B-stamp $\odot_{i\in I}\mathsf{R}_i = (A, \mu, M, p, X)$ where $\mu\colon w \mapsto (\mu_i(w))_{i\in I}$ is the surjective co-restriction of the product map and (M, p, X) is the B-submonoid given by the image of μ in the product of the (M_i, p_i, X_i)'s. This construction yields a B-stamp, the product of BiM presentations is a BiM presentation, but the product of typed stamps may not be a typed stamp. Thus we define the *restricted product with respect to a finite subset* $F \subseteq I$ to be the B-stamp obtained from the product by composing $p\colon M \to X$ with the restriction of $\prod_{j\in F} \pi_j$ where $\pi_j\colon \Pi X_i \to X_j$ are the projections. Note that every restricted product of typed stamps is a typed stamp.

Let M be a monoid and $\mathcal{S} \subseteq \mathcal{T}_M$. We say $\mathcal{S}$ is *separating* provided whenever $m_1, m_2 \in M$ are distinct, there is $(M, p, X) \in \mathcal{S}$ such that $p(m_1) \neq p(m_2)$. Finally, for a monoid presentation (A, μ, M) and a family of typed stamp $\mathcal{F}$, denote by $\mathsf{Typ}_{\mathcal{F}}(A, \mu, M)$ the set of all typed monoids (M, p, X) such that (A, μ, M, p, X) belongs to $\mathcal{F}$. A monoid quotient $g\colon M \twoheadrightarrow N$ is said to be *$\mathcal{F}$-compatible* provided $\{(N, q, Y) \in \mathcal{T}_N \mid (M, q \circ g, Y) \in \mathsf{Typ}_{\mathcal{F}}(A, \mu, M)\}$ is separating. A stamp $\mathsf{S} = (A, \nu, N, q, Y)$ is an *$\mathcal{F}$-compatible quotient* of $\mathsf{R} = (A, \mu, M, p, X)$ provided S factors through R and the corresponding quotient map $M \twoheadrightarrow N$ is $\mathcal{F}$-compatible.

▶ **Definition 8.** A family of typed stamps V is called a *$\mathcal{C}$-pseudovariety* provided

(V.1) V is closed under taking $\mathcal{C}$-substamps and V-compatible quotients;

(V.2) V is closed under taking arbitrary restricted products;

(V.3) $\mathsf{Typ}_V(A, \mu, M)$ is a multiplicative and separating downset whenever (A, μ, M) is a monoid presentation of a typed stamp in V.

We are now able to state the main result of this section.

▶ **Theorem 9.** *There is a one-to-one correspondence between $\mathcal{C}$-varieties of languages and $\mathcal{C}$-pseudovarieties of typed stamps.*

Proof Sketch. The correspondence is given as follows: for each $\mathcal{C}$-pseudovariety of typed stamps V, we define $\mathcal{V}(A)$ to be the set of all languages over A that are recognised by some element of V. Conversely, given a $\mathcal{C}$-variety of languages $\mathcal{V}$, for each finite alphabet A, the syntactic BiM of $\mathcal{V}(A)$ is a BiM presentation R_A. In turn, the BiM presentations that factor through R_A are given by projective limit systems of typed stamps. Altogether, these form a $\mathcal{C}$-pseudovariety. Finally one can show that these two constructions are inverse to each other. ◀

Observe that, unlike what happens for classical stamps on finite monoids, not all classes of typed stamps generate $\mathcal{C}$-pseudovarieties. For this to be the case, the multiplicative closure of the class has to be separating for each of the monoid presentations of the class.

3.4 Using profinite alphabets

Let V be a $\mathcal{C}$-pseudovariety of typed stamps, and $\mathcal{V}$ the corresponding $\mathcal{C}$-variety of languages. For each finite alphabet A, the class of typed stamps in V based on A forms a projective limit system whose projective limit is a BiM presentation, which we will denote by $\Sigma_A(V)$. As indicated by the proof sketch for Theorem 9, the point is that $\Sigma_A(V)$ is also the syntactic B-stamp of $\mathcal{V}(A)$. In case $\mathcal{V}$ consists entirely of regular languages, $\Sigma_A(V)$ is essentially what is usually denoted by $\overline{\Omega}_A(V_{\mathrm{mon}})$ or $\hat{F}_A(V_{\mathrm{mon}})$ and is called the *free A-generated pro-*$\mathrm{V}_{\mathrm{mon}}$ *monoid*, where V_{mon} is the pseudovariety of all finite monoids M for which (M, id, M) is the B-monoid component of a stamp in V.

As in the setting of regular languages, it is sometimes useful to extend varieties to profinite alphabets. This will be crucial for our main result, Theorem 23. Let Y be a profinite alphabet and let $\{\pi_i : Y \twoheadrightarrow Y_i\}_{i\in I}$ be the set of all finite continuous quotients of Y. Further let $\{h_{i,j} : Y_i \twoheadrightarrow Y_j \mid \pi_j = h_{i,j} \circ \pi_i\}$ be the diagram of all quotient maps commuting with the projections. Then this is a projective limit system, and it is a well-known fact that Y is the projective limit of this system (see Appendix A). On the other hand, since $\mathcal{V}$ is a $\mathcal{C}$-variety, each of the maps $h_{i,j}$ defines a unique monoid quotient map $h^*_{i,j} : Y_i^* \twoheadrightarrow Y_j^*$, which dually defines an embedding of Boolean algebras $(h^*_{i,j})^{-1} : \mathcal{V}(Y_j) \hookrightarrow \mathcal{V}(Y_i)$. Therefore, using a slight adaptation of Proposition 7, we have that there exists a morphism of the form $\Phi_{i,j} = (h_{i,j}, g_{i,j}, \varphi_{i,j})$ from $\Sigma_{Y_i}(V)$ to $\Sigma_{Y_j}(V)$. Thus, the family $\{\Sigma_{Y_i}(V)\}_{i\in I}$ defines a projective system with connecting morphisms $\Phi_{i,j}$.

▶ **Proposition 10.** *The projective limit of the family $\{\Sigma_{Y_i}(\mathrm{V})\}_{i\in I}$ exists and it is a* BiM *presentation (Y, μ, M, p, X) on the profinite alphabet Y, which we denote by $\Sigma_Y(\mathrm{V})$.*

4 Logic on Words

Logic on words (see, e.g. [15]), is a logical language whose intended models are *words*, that is, elements of the free monoid A^*, for a fixed finite alphabet A. We shall consider formulas that are recursively built as follows:

- *letter predicates* are formulas: for each $a \in A$, we have a letter predicate $\mathsf{P}_a(x)$;
- *numerical predicates* are formulas: given $k \geq 0$, a k-ary numerical predicate is given by a relation R that assigns to each element n of $\mathbb{N}$ a subset R_n of $\{1, \ldots, n\}^k$ (for instance, the (binary) numerical predicate $x < y$ assigns to each n the set $\{(i, j) \mid 1 \leq i < j \leq n\}$);
- *Boolean combinations* of formulas are formulas: if φ and ψ are formulas, then so are $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\neg\varphi$;
- *unary quantification* of a formula with respect to a variable is a formula: a quantifier is given by a map $\mathsf{Q} : \{0, 1\}^* \to \{0, 1\}$ (for instance, the existential quantifier $\exists$ maps the word $\varepsilon_1 \cdots \varepsilon_k \in \{0, 1\}^*$ to 1 if and only if there exists an index i such that $\varepsilon_i = 1$).

Sentences with letter predicates from A have words of A^* as models. More generally, models of formulas with free variables in $\Upsilon = \{x_1, \ldots, x_k\}$ are given by Υ-*structures*, $w_{x_1=i_1,\ldots,x_k=i_k}$, where $w \in A^*$ and $i_1, \ldots, i_k \in \{1, \ldots, |w|\}$ (cf. [15, Chapter II]). We denote by $A^* \otimes \Upsilon$ the set of all Υ-structures. The semantics of a formula can then be defined inductively as follows. The $\{x\}$-structure $w_{x=i}$ satisfies $\mathsf{P}_a(x)$ if and only if its i-th letter is an a, and $w_{x_1=i_1,\ldots,x_k=i_k}$ satisfies the atomic formula $R(x_1, \ldots, x_k)$, where R is a k-ary numerical predicate, if and only if $(i_1, \ldots, i_k)$ belongs to $R_{|w|}$. The Boolean connectives *and* $\wedge$, *or* $\vee$, and *not* $\neg$ are interpreted classically. A *quantifier* $\mathsf{Q} : \{0, 1\}^* \to \{0, 1\}$ is interpreted as follows: w satisfies $\mathsf{Q}x\ \varphi(x)$ if and only if $\mathsf{Q}(\delta_{\varphi(x)}(w)) = 1$, where $\delta_{\varphi(x)} : A^* \to \{0, 1\}^*$ sends $w = a_1 \cdots a_k$ to $\varepsilon_1 \cdots \varepsilon_k$ where $\varepsilon_i = 1$ if and only if $w_{x=i}$ satisfies $\varphi(x)$. Important

examples of quantifiers are given by the *existential quantifier* $\exists$ mentioned above, *modular quantifiers* $\exists^r_q$, for each $q \in \mathbb{N}$ and $0 \leq r < q$, mapping a word of $\{0,1\}^*$ to 1 if and only if the number of 1's is congruent to r modulo q, and the *majority quantifier* Maj which sends an element of $\{0,1\}^*$ to 1 if and only if it has strictly more occurrences of 1 than of 0. Finally, for a fixed set of free variables Υ, given a formula φ with free variables in Υ, we denote by L_φ the set of all Υ-structures satisfying φ.

In what follows, we fix a logical signature with set of numerical predicates $\mathcal{N}$ and set of unary quantifiers $\mathcal{Q}$. For each finite alphabet A and each finite set of variables Υ, we denote by $\mathcal{Q}_A[\mathcal{N}](\Upsilon)$ the corresponding set of first-order formulas with free variables in Υ, and letter predicates for $a \in A$. For reasons which will become apparent in Section 4.1, we are interested in studying fragments of logic allowing the alphabet to vary.

▶ **Definition 11.** A *logic class* Γ is a map that associates to each finite alphabet A and finite set Υ of first-order variables, a set of formulas $\Gamma_A(\Upsilon) \subseteq \mathcal{Q}_A[\mathcal{N}](\Upsilon)$ which satisfies the following properties:

(LC.1) If $\Upsilon \subseteq \Upsilon'$, then $\Gamma_A(\Upsilon) \subseteq \Gamma_A(\Upsilon')$;

(LC.2) Each set $\Gamma_A(\Upsilon)$ is closed under Boolean connectives $\wedge$ and $\neg$ (and thus, under $\vee$);

(LC.3) Every map $\zeta\colon A \to B$ between finite alphabets induces a map $\zeta_{\Gamma,\Upsilon} : \Gamma_B(\Upsilon) \to \Gamma_A(\Upsilon)$ which sends $\varphi \in \Gamma_B(\Upsilon)$ to the formula obtained by substituting, for every occurrence in φ of a predicate $\mathsf{P}_b(x)$ with $b \in B$, the formula $\bigvee_{\zeta(a)=b} \mathsf{P}_a(x)$. Note that if b is not in the image of ζ, then $\mathsf{P}_b(x)$ is replaced by the empty join, which is logically equivalent to the *always-false* proposition.

The intuitive idea behind the definition of a logic class is to model what is usually referred to as a *fragment of logic*. For instance, considering all formulas of quantifier depth less than k in a given logic defines a logic class.

We consider formulas up to semantic equivalence, and thus, by (LC.2), each $\Gamma_A(\Upsilon)$ is a Boolean algebra. The associated partial order relation is given by semantic implication: $\varphi \leq \psi$ if and only if $L_\varphi \subseteq L_\psi$. That is, $\Gamma_A(\Upsilon)$ is isomorphic to the Boolean algebra of languages $\{L_\varphi \mid \varphi \in \Gamma_A(\Upsilon)\}$.

Restricting to sentences, we obtain a language class $A \mapsto \mathcal{L}_A(\Gamma) = \{L_\varphi \mid \varphi \in \Gamma_A(\emptyset)\}$. Note that property (LC.3) of logic classes implies that the associated class of languages is closed under inverse images of *lp*-morphisms. Therefore, if each $\mathcal{L}_A(\Gamma)$ is closed under quotienting by words, then the language class is (at least) an *lp*-variety.

4.1 Substitution

The concept of substitution for the study of logic on words, as in [17], is quite different from substitution in predicate logic. Substitution in predicate logic works on terms, whereas the notion of substitution in [17] works at the propositional level of the predicate logic. As such it provides a method for decomposing complex formulas into simpler ones. The core idea is to enrich the alphabet over which the logic is defined in order to be able to substitute large subformulas through letter predicates.

Roughly speaking, *substitution* is a tool for decomposing formulas into simpler ones. For instance, the sentence $\psi = \exists x\ \varphi(x)$ may be obtained from the sentence $\exists x\ \mathsf{P}_b(x)$ by replacing $\mathsf{P}_b(x)$ by $\varphi(x)$. Then, understanding ψ amounts to understanding both the sentence $\exists x\ \mathsf{P}_b(x)$ and the formula $\varphi(x)$. If we want to substitute away several subformulas in this way, we must account for their logical relations. The idea of an alphabet is that the corresponding predicates $\{\mathsf{P}_a(x)\}_{a\in A}$ interpret in any word as a finite set $\mathcal{F}$ of formulas satisfying:

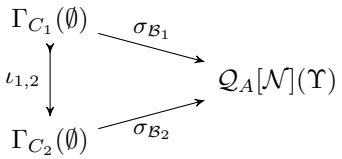
(A.1) $\vee_{\varphi\in\mathcal{F}}\ \varphi$ is the *always-true* proposition;

(A.2) for every $\varphi_1, \varphi_2 \in \mathcal{F}$ distinct, $\varphi_1 \wedge \varphi_2$ is the *always-false* proposition.

We formalise this concept of substitution. Suppose we are given two logic classes Λ and Γ.[1] Further let C be a finite alphabet and $\xi : C \to \Lambda_A(\Upsilon \cup \{x\})$, where $x \notin \Upsilon$, a map whose image satisfies (A.1) and (A.2). Note that, this is equivalent to requiring that the formulas in the image of ξ are precisely the atoms of the Boolean algebra $\mathcal{B}$ generated by the image of ξ. Now we may define the *substitution* given by ξ to be the map $\sigma_\xi : \Gamma_C(\emptyset) \to \mathcal{Q}_A[\mathcal{N}](\Upsilon)$ defined by substituting for any occurrence of a letter predicate $\mathsf{P}_c(z)$ in a sentence $\psi \in \Gamma_C(\emptyset)$, the formula $\xi(c)(x/z)$ (that is, the formula obtained by substituting z for x in the formula $\xi(c) \in \Lambda_A(\Upsilon \cup \{x\})$). Note that here we assume (without loss of generality) that things have been arranged so that the variables occurring in ψ, such as z, do not occur in the formulas in the image of ξ. Since the only constraints on the interpretation of letters in a word are given by the properties (A.1) and (A.2), it follows by a simple structural induction that σ_ξ is a morphism of Boolean algebras. We denote the image of this morphism by $\Gamma \circ \mathcal{B}$.

Note this name makes sense as $\Gamma \circ \mathcal{B}$ is uniquely determined by $\mathcal{B}$. Indeed, instead of starting from a map ξ whose image satisfies (A.1) and (A.2), we could start with a finite Boolean subalgebra $\mathcal{B}$ of $\Lambda_A(\Upsilon \cup \{x\})$. Then we obtain a finite alphabet by letting $C_\mathcal{B} = \mathsf{At}(\mathcal{B})$ and a map $\xi_\mathcal{B}$ given by the inclusion of $\mathsf{At}(\mathcal{B})$ in $\Lambda_A(\Upsilon \cup \{x\})$. The resulting substitution, which we will denote by $\sigma_\mathcal{B}$ (instead of $\sigma_{\xi_\mathcal{B}}$), has $\Gamma \circ \mathcal{B}$ as its image. With this notation in place, we can now compare the substitution maps obtained for different finite Boolean subalgebras of $\Lambda_A(\Upsilon \cup \{x\})$. In fact, in what follows, we shall argue that the concept of substitution is naturally extendable to infinite Boolean algebras.

Let $\mathcal{B}_1 \subseteq \mathcal{B}_2$ be finite Boolean subalgebras of $\Gamma_A(\Upsilon \cup \{x\})$. Then the dual of the inclusion map $\mathcal{B}_1 \hookrightarrow \mathcal{B}_2$ sends each atom of $\mathcal{B}_2$ to the unique atom of $\mathcal{B}_1$ that is above it (in the order on $\mathcal{B}_2$). This yields a map $\xi_{1,2} : C_2 \twoheadrightarrow C_1$, where $C_i = \mathsf{At}(\mathcal{B}_i)$, for $i = 1, 2$, are the corresponding alphabets. Now since Γ is a language class, by property (LC.3), $\xi_{1,2}$ yields a morphism $\iota_{1,2} \colon \Gamma_{C_1}(\emptyset) \to \Gamma_{C_2}(\emptyset)$. Moreover, since every element of $\mathcal{B}_1$ is logically equivalent to the disjunction of the atoms of $\mathcal{B}_2$ below it, the morphism $\iota_{1,2}$ is in fact an embedding and the following diagram commutes:



That is, $\mathcal{B}_1 \subseteq \mathcal{B}_2$ yields $\Gamma \circ \mathcal{B}_1 \subseteq \Gamma \circ \mathcal{B}_2$. Thus we have a direct system of subalgebras of $\mathcal{Q}_A[\mathcal{N}](\Upsilon)$ (see Appendix A). This allows us to extend the composition to infinite Boolean algebras of formulas and thereby also to language classes.

▶ **Definition 12.** Let Λ and Γ be logic classes. For each finite set Υ of variables, and each finite alphabet A, we define

$$(\Gamma \circ \Lambda)_A(\Upsilon) = \left\langle \varinjlim \{\Gamma \circ \mathcal{B} \mid \mathcal{B} \subseteq \Lambda_A(\Upsilon \cup \{x\}) \text{ is a finite Boolean algebra}\} \cup \Lambda_A(\Upsilon) \right\rangle_{\mathsf{BA}},$$

where the connecting morphisms are the inclusions $\Gamma \circ \mathcal{B}_1 \subseteq \Gamma \circ \mathcal{B}_2$ whenever $\mathcal{B}_1 \subseteq \mathcal{B}_2$.

▶ **Proposition 13.** *Let Λ and Γ be logic classes. Then, $\Gamma \circ \Lambda$ is also a logic class.*

[1] We will only use the sentences of Γ, so we may assume that Γ consists entirely of sentences. It may for example be all the depth one sentences using some quantifier of interest.

4.2 Substitution and Duality: The Substitution Principle

Substitution allows us to describe logic classes as compositions of simpler ones. In Section 5 we treat recognition for languages given by formulas obtained by substitution in terms of recognisers of the components. Key ingredients for this are the dual of the substitution map, which we describe in Proposition 14, and the concrete description of the languages given by formulas obtained by substitution (Corollary 15) that it provides.

Let Γ and Λ be logic classes and A a finite alphabet. In this subsection, our goal is to describe the languages in $\mathcal{L}_A(\Gamma \circ \Lambda)$, or equivalently, the languages defined by sentences of $(\Gamma \circ \Lambda)_A(\emptyset)$. If $\mathcal{B}$ is a finite Boolean subalgebra of $\Lambda_A(x)$ and $C_\mathcal{B}$=$\mathsf{At}(\mathcal{B})$, then $\mathcal{B}$ defines a substitution morphism $\sigma_\mathcal{B}\colon \Gamma_{C_\mathcal{B}}(\emptyset) \to (\Gamma \circ \Lambda)_A(\emptyset)$, which may be seen as a morphism $\sigma_\mathcal{B}\colon \mathcal{L}_{C_\mathcal{B}}(\Gamma) \to \mathcal{L}_A(\Gamma \circ \Lambda)$ between the corresponding Boolean algebras of languages given by $L_\psi \mapsto L_{\sigma_\mathcal{B}(\psi)}$. Let $\Sigma_\mathcal{B}$ be the Stone dual of $\sigma_\mathcal{B}$. Since $\mathcal{L}_A(\Gamma \circ \Lambda)$ and $\mathcal{L}_{C_\mathcal{B}}(\Gamma)$ are Boolean algebras of languages over A^* and $C_\mathcal{B}^*$, respectively, we have maps $p\colon A^* \to X_{\mathcal{L}_A(\Gamma\circ\Lambda)}$ and $q\colon C_\mathcal{B}^* \to X_{\mathcal{L}_{C_\mathcal{B}}(\Gamma)}$ with dense images obtained as the restrictions of the dual maps of the inclusions $\mathcal{L}_A(\Gamma \circ \Lambda) \hookrightarrow \mathcal{P}(A^*)$ and $\mathcal{L}_{C_\mathcal{B}}(\Gamma) \hookrightarrow \mathcal{P}(C_\mathcal{B}^*)$, respectively. A continuous map $X_{\mathcal{L}_A(\Gamma\circ\Lambda)} \to X_{\mathcal{L}_{C_\mathcal{B}}(\Gamma)}$ need not restrict to the dense images of the monoids, however this is indeed the case for the maps $\Sigma_\mathcal{B}$. This is a consequence of the following stronger result.

▶ **Proposition 14.** *Let $\mathcal{B}$ be a finite Boolean subalgebra of $\Lambda_A(x)$, $C_\mathcal{B} = \mathsf{At}(\mathcal{B})$, and let $\xi\colon A^* \otimes \{x\} \to C_\mathcal{B}$ be the dual to the embedding $\mathcal{B} \hookrightarrow \mathcal{P}(A^* \otimes \{x\})$. Then, the function $\tau_\mathcal{B} : A^* \to C_\mathcal{B}^*$ defined by $\tau(w) = \xi(w_{x=1}) \cdots \xi(w_{x=|w|})$ makes the diagram commute:*

$$\begin{array}{ccc} A^* & \xrightarrow{\tau_\mathcal{B}} & C_\mathcal{B}^* \\ {\scriptstyle rp}\downarrow & & \downarrow{\scriptstyle q} \\ X_{\mathcal{L}_A(\Gamma\circ\Lambda)} & \xrightarrow{\Sigma_\mathcal{B}} & X_{\mathcal{L}_{C_\mathcal{B}}(\Gamma)} \end{array}$$

Recall, by Definition 12, that $(\Gamma \circ \Lambda)_A(\emptyset)$ is generated as a Boolean algebra by $\Lambda_A(\emptyset)$ and the sentences of the form $\sigma_\mathcal{B}(\psi)$ for $\psi \in \Gamma_{C_\mathcal{B}}(\emptyset)$ for $\mathcal{B}$ a finite Boolean subalgebra of $\Lambda_A(x)$. From the proof of Proposition 14 (see (4) in Appendix C.2), we obtain a concrete description of the languages corresponding to $\mathcal{L}_A(\Gamma \circ \Lambda)$.

▶ **Corollary 15** (Substitution Principle)**.** *The Boolean algebra $\mathcal{L}_A(\Gamma \circ \Lambda)$ is generated by the languages of $\mathcal{L}_A(\Lambda)$ together with the languages of the form $\tau_\mathcal{B}^{-1}(K)$, where $\mathcal{B} \subseteq \Lambda_A(x)$ is a finite Boolean subalgebra and $K \in \mathcal{L}_{C_\mathcal{B}}(\Gamma)$.*

Dualising the direct limit system $\{\Gamma \circ \mathcal{B}\}_\mathcal{B}$ discussed in Section 4.1, we obtain a projective system (see Appendix A):

▶ **Proposition 16.** *The class of morphisms*

$$\{\tau_\mathcal{B} : A^* \to C_\mathcal{B}^* \mid \mathcal{B} \subseteq \Lambda_A(x) \text{ is a finite Boolean subalgebra}\}$$

forms a projective limit system, where the connecting morphisms are the morphisms of monoids $C_{\mathcal{B}_2}^* \to C_{\mathcal{B}_1}^*$ induced by the dual maps of inclusions $\mathcal{B}_1 \hookrightarrow \mathcal{B}_2$. The limit of this system is the map $\tau : A^* \to X_{\Lambda_A(x)}^*$ sending a word $w \in A^*$ to the word $\mu_1 \cdots \mu_{|w|}$ with $\mu_i = \{\varphi(x) \in \Lambda_A(x) \mid w_{x=i} \text{ satisfies } \varphi(x)\}$, and $X_{\Lambda_A(x)}$ is the dual of $\Lambda_A(x)$.

Note that the maps $\tau_\mathcal{B}$ are all length preserving, thus the above system factors into projective limit systems $\{\tau_\mathcal{B} : A^n \to (C_\mathcal{B})^n \mid \mathcal{B} \subseteq \Lambda_A(x)$ is a finite Boolean subalgebra$\}$ for each $n \in \mathbb{N}$, and each of these systems has a profinite limit in the usual sense. The space $X_{\Lambda_A(x)}^*$ obtained in Proposition 16 is then seen as the union over $n \in \mathbb{N}$ of the spaces $X_{\Lambda_A(x)}^n$, each one of those being a Boolean space when equipped with the product topology.

5 The Block Product Principle

It is well-known that when the logic classes considered define only regular languages, the decomposition of first-order formulas given by the Substitution Principle is modelled by the usual block product of finite monoids (see [17] for a survey). In this section, we consider a notion of block product that generalises the usual one from a logic perspective, thereby obtaining a Block Product Principle for *lp*-pseudovarieties of typed stamps (cf. Theorem 21).

The Substitution Principle justifies the interest in studying the family of maps $\tau_{\mathcal{B}}$ parameterised by finite Boolean subalgebras $\mathcal{B} \subseteq \Lambda_A(x)$. In turn, each language defined by a formula $\varphi(x)$ may be seen as a language over the extended alphabet $A \times 2^{\{x\}}$, and therefore it is recognised by some typed stamp over the alphabet $A \times 2^{\{x\}}$. The following notion of *transduction* encodes the maps $\tau_{\mathcal{B}}$ in terms of recognition (cf. Proposition 18).

▶ **Definition 17.** Let $\mathsf{S} = (A \times 2^{\{x\}}, \nu, N, q, Y)$ be a typed stamp. We let C_{S} be the alphabet contained in Y given by the image $q \circ \nu[A^* \otimes \{x\}]$. The *transduction determined by* S is the map $\tau_{\mathsf{S}} : A^* \to C_{\mathsf{S}}^*$ given by $\tau_{\mathsf{S}}(w) = q \circ \nu(w_{x=1}) \cdots q \circ \nu(w_{x=|w|})$.

▶ **Proposition 18.** *Let $\mathcal{B} \subseteq \Lambda_A(x)$ be a finite Boolean algebra, and $\mathsf{S} = (A \times 2^{\{x\}}, \nu, N, q, Y)$ be the syntactic typed stamp of the Boolean subalgebra of $\mathcal{P}((A \times 2^{\{x\}})^*)$ generated by the set of languages definable by a formula of $\mathcal{B}$. Then, C_{S} is isomorphic as a set to $C_{\mathcal{B}} = \mathsf{At}(\mathcal{B})$, and $\tau_{\mathsf{S}} = \tau_{\mathcal{B}}$.*

Now, in view of the Substitution Principle and of Proposition 18, our goal is to identify the recognisers of languages of the form $\tau_{\mathsf{S}}^{-1}(K)$ when $K \subseteq C_{\mathsf{S}}^*$ is recognised by a given typed stamp R. The block product $\mathsf{R} \,\square\, \mathsf{S}$, that we introduce in Definition 19 below, has the property of recognising these languages and little more (cf. Proposition 20).

Recall that, given monoids M and N, their *block product*, $M \,\square\, N$, is the monoid with underlying set $M^{N \times N} \times N$ and binary operation given by $(f, n)(f', n') = (h, n \cdot n')$, where $h(n_1, n_2) = f(n_1, n' \cdot n_2) + f'(n_1 \cdot n, n_2)$. Here, in order to improve readability, we are denoting the operation on M additively and that on N multiplicatively, although neither is assumed to be commutative.

▶ **Definition 19.** Let $\mathsf{R} = (C_{\mathsf{S}}, \mu, M, p, X)$ and $\mathsf{S} = (A \times 2^{\{x\}}, \nu, N, q, Y)$ be typed stamps. For each letter $a \in A$, we define the function $f_a : N \times N \to M$ by

$$f_a(n_1, n_2) = \mu \circ q(n_1 \, \nu(a, \{x\}) \, n_2),$$

and we let $\mathcal{F}_{\mathsf{R},\mathsf{S}}$ be the set of all such functions. Then, the *block product* of R and S, denoted $\mathsf{R} \,\square\, \mathsf{S}$, is the typed stamp

$$(A, \mu \,\square_q\, \nu, \langle \mathcal{F}_{\mathsf{R},\mathsf{S}} \times \nu(A) \rangle, p \,\square\, q, X \times Y),$$

where $\langle \mathcal{F}_{\mathsf{R},\mathsf{S}} \times \nu(A) \rangle$ is the submonoid of the usual block product of monoids $M \,\square\, N$ generated by $\mathcal{F}_{\mathsf{R},\mathsf{S}} \times \nu(A)$, $\mu \,\square_q\, \nu$ is the unique homomorphism mapping $a \in A$ to $(f_a, \nu(a))$, and $p \,\square\, q$ sends (f, n) to $(p \circ f(1, 1), q(n))$.

We now state the local version of the *Block Product Principle*.

▶ **Proposition 20.** *Let $\mathsf{R} = (C_{\mathsf{S}}, \mu, M, p, X)$ and $\mathsf{S} = (A \times 2^{\{x\}}, \nu, N, q, Y)$ be typed stamps. Then, a language $L \subseteq A^*$ is recognised by $\mathsf{R} \,\square\, \mathsf{S}$ if and only if it is a Boolean combination of languages of the form $\tau_{\mathsf{S}}^{-1}(K_1) \cap K_2$ for some language $K_1 \subseteq C_{\mathsf{S}}^*$ recognised by R and some language $K_2 \subseteq A^*$ recognised by the lp-substamp of S defined by the injective map $A \hookrightarrow A \times 2^{\{x\}}$ sending a to $(a, \emptyset)$.*

Given *lp*-pseudovarieties V and W of typed stamps, we denote by $V\square W$ the *lp*-pseudovariety generated by $\mathsf{R}\square\mathsf{S}$ for $\mathsf{R}\in V$ and $\mathsf{S}\in W$ typed stamps over suitable alphabets. Note that, the multiplicative closure of this set of block products forms a separating set for each of the monoid presentations. We denote by $\mathcal{V}\boxplus\mathcal{W}$ the corresponding *lp*-variety of languages (recall Theorem 9). As a consequence of Proposition 20 we obtain:

▶ **Theorem 21** (Block Product Principle). *Let* V *and* W *be lp-pseudovarieties of typed stamps. Then, the Boolean algebra* $(\mathcal{V}\boxplus\mathcal{W})(A)$ *is generated by the languages of* $\mathcal{W}(A)$ *together with the languages of the form* $\tau^{-1}(K)$*, where* τ *is a transduction defined by an element of* W *and* K *belongs to* $\mathcal{V}(C)$ *for a suitable alphabet* C*.*

6 A generalisation of Semidirect Product

In Section 5, we showed that the block product of typed stamps is suitable for the recognition of languages defined by formulas obtained by substitution. In this section, we describe the structure of the syntactic B-stamp $\Sigma_A(V\square W)$ of the Boolean algebra of languages recognised by some element of $V\square W$, when V and W are two given *lp*-pseudovarieties (cf. Theorem 23). This may be considered as analogue to the result of Almeida and Weil [1] describing the A-generated free pro-$(V{*}{*}W)$ monoid as a two-sided semidirect product of the free pro-V and pro-W monoids over suitable alphabets. In their result, V and W are pseudovarieties of finite monoids in the usual sense (recall the beginning of Section 3.4) and $V{*}{*}W$ denotes the pseudovariety generated by the (usual) block product of monoids, $M\square N$, for $M\in V$ and $N\in W$.

Let M and N be monoids. Again, we denote the operation on M additively and that on N multiplicatively.

A *monoid biaction of* N *on* M is a biaction of N on the underlying set of M, satisfying the following additional properties:

- $n_1\cdot(m_1+m_2)\cdot n_2 = n_1\cdot m_1\cdot n_2 + n_1\cdot m_2\cdot n_2$, for all $n_1,n_2\in N$, and $m_1,m_2\in M$;
- $n_1\cdot 0\cdot n_2 = 0$, for all $n_1,n_2\in N$.

Given such a biaction, we may define a new monoid, called the *two-sided semidirect product*, usually denoted by $M{*}{*}N$. It has underlying set $M\times N$, and operation defined by

$$(m_1,n_1)(m_2,n_2) = (m_1\cdot n_2 + n_1\cdot m_2, n_1n_2).$$

An example of this construction is given by the usual block product of monoids.

More generally, if $\mathsf{R}=(M,p,X)$ and $\mathsf{S}=(N,q,Y)$ are B-monoids and N bi-acts on M, one may define the *two-sided semidirect product* of R and S to be the B-monoid $(M{*}{*}N, p\times q, X\times Y)$ where $M{*}{*}N$ is the usual semidirect product of monoids with respect to the given biaction. Note that, even if R and S are both BiM's, the resulting semidirect product need not be a BiM. Let $\Sigma_{A\times 2}(W)=(A\times 2^{\{x\}},\nu,N,q,Y)$ and $\Sigma_Y(V)=(Y,\mu,M,p,X)$. In order to understand the structure of the BiM component of $\Sigma_A(V\square W)$, we start by showing that N naturally bi-acts on X. Intuitively, that is a consequence of $\mathcal{V}(C)$ being closed under quotients for every finite alphabet C.

▶ **Proposition 22.** *There is a biaction of* N *on* X *whose left and right components at each* $n\in N$*,* $\psi_{\ell,n}$ *and* $\psi_{r,n}$*, respectively, are continuous maps that satisfy*

$$\psi_{\ell,n}\circ p\circ\mu(y_1,\dots,y_k) = p\circ\mu(\lambda_n(y_1),\dots,\lambda_n(y_k)), \tag{1}$$

$$\psi_{r,n}\circ p\circ\mu(y_1,\dots,y_k) = p\circ\mu(\rho_n(y_1),\dots,\rho_n(y_k)). \tag{2}$$

In particular, N bi-acts on M and hence, there is a well-defined two-sided semidirect product $(M,p,X)**(N,q,Y)$. Moreover, each homomorphism $h : C^* \to M**N$ defines a B-stamp $\Sigma_Y(V)**\Sigma_{A\times 2}(W)$ over the alphabet C, whose B-monoid component is a B-submonoid of $(M,p,X)**(N,q,Y)$. We are now able to state the main result of this section.

▶ **Theorem 23.** *Let* V *and* W *be lp-pseudovarieties of typed stamps. Then, the* BiM *presentation* $\Sigma_A(\mathrm{V}\,\square\,\mathrm{W})$ *is isomorphic to a two-sided semidirect product* $\Sigma_Y(\mathrm{V})**\Sigma_{A\times 2}(\mathrm{W})$.

Proof Sketch. Let $\Sigma_A(V\square W) = (A,\eta,P,r,Z)$, and let $\tau : A^* \to Y^*$ be the map given by

$$\tau(w) = (q\circ\nu(w_{x=1}),\dots,q\circ\nu(w_{x=|w|})),$$

for $w \in A^*$. In other words, the co-restriction of τ to its image is the limit of the projective system described in Proposition 16. We claim that the homomorphism $h : A^* \to M**N$ sending the word w to the pair $(\mu\circ\tau(w),\nu(w))$ defines a semidirect product $\Sigma_Y(V)**\Sigma_{A\times 2}(W)$, which is isomorphic to $\Sigma_A(V\square W)$.

Using the Block Product Principle, we may prove the existence of an onto morphism of Boolean algebras $\mathsf{Clopen}(X)\oplus\mathsf{Clopen}(Y) \twoheadrightarrow (\mathcal{V}\square\mathcal{W})(A)$, which dually defines an embedding of Boolean spaces $\theta : Z \hookrightarrow X\times Y$. On the other hand, one may also prove the inclusion $\ker(\eta)\subseteq\ker(h)$, and thus, there is a homomorphism $g : P \to M**N$ satisfying $g\circ\eta = h$ and $\theta\circ r = (p\times q)\circ g$. In particular, g is necessarily injective and this proves the claim. ◀

7 Applications to logic

In this section, we show how to apply the results of the paper to decompose a logic class $\mathcal{Q}[\mathcal{N}]$ under the assumption that it admits a prenex normal form. That is, each formula is equivalent to one in which a string of quantifiers is applied to a quantifier free formula. Requiring a logic signature to admit prenex normal forms is a mild condition which is satisfied as soon as the logical language is sufficiently expressive. Indeed, this is the case for many standard classes considered in Boolean circuit complexity, such as those referred to in the introduction.

Given a set of quantifiers $\mathcal{Q}$, we let $\Gamma_\mathcal{Q}$ be the logic class of sentences assigning to the finite alphabet A the Boolean algebra generated by formulas of the form $\mathsf{Q}x\ \bigvee_{a\in B}\mathsf{P}_a(x)$, with $\mathsf{Q}\in\mathcal{Q}$ and $B\subseteq A$. In turn, if $\mathcal{N}$ is a set of numerical predicates, then $\Lambda_\mathcal{N}$ is the logic class in which $\Lambda_{\mathcal{N},A}(\Upsilon)$ consists of all Boolean combinations of numerical predicates from $\mathcal{N}$, letter predicates for letters in A, and having free variables in Υ.

Suppose we are given a finite Boolean subalgebra $\mathcal{B}\subseteq\Lambda_{\mathcal{N},A}(\{x_1,\dots,x_{k-1},x_k\})$. The substitution map defined by $\mathcal{B}$ in Section 4.1 has image $\Gamma_\mathcal{Q}\circ\mathcal{B}$, which is a finite Boolean subalgebra of $(\Gamma_\mathcal{Q}\circ\Lambda_\mathcal{N})_A(\{x_1,\dots,x_{k-1}\})$ (when we take $\Upsilon = \{x_1,\dots,x_{k-1}\}$). We may now consider the substitution map defined by $\Gamma_\mathcal{Q}\circ\mathcal{B}$, thereby obtaining a finite Boolean subalgebra of $(\Gamma_\mathcal{Q}\circ(\Gamma_\mathcal{Q}\circ\Lambda_\mathcal{N}))_A(\{x_1,\dots,x_{k-2}\})$. By successively iterating the operation $\Lambda\mapsto(\Gamma_\mathcal{Q}\circ\Lambda)$, one is able to produce all the prenex normal form sentences of $\mathcal{Q}_A[\mathcal{N}](\emptyset)$ of a given quantifier depth. Thus we obtain:

▶ **Proposition 24.** *Let* $\mathcal{Q}$ *be a set of quantifiers and* $\mathcal{N}$ *a set of numerical predicates such that every formula of* $\mathcal{Q}_A[\mathcal{N}]$ *admits a prenex normal form. Then,*

$$\mathcal{Q}_A[\mathcal{N}](\emptyset) = \bigcup_{n\in\mathbb{N}} \underbrace{\Gamma_\mathcal{Q}\circ(\cdots\circ(\Gamma_\mathcal{Q}}_{n\ times}\circ\Lambda_\mathcal{N})\dots)_A(\emptyset).$$

In order to be able to apply the results of previous section and, through Proposition 24, obtain recognisers for the languages definable in $\mathcal{Q}[\mathcal{N}]$, one should identify the *lp*-pseudovarieties of typed stamps that recognise the languages definable in the logic classes $\Gamma_\mathcal{Q}$ and $\Lambda_\mathcal{N}$.

▶ **Proposition 25.** *Given a quantifier* $\mathsf{Q} : \{0,1\}^* \to \{0,1\}$*, we let* L_{Q} *be the language* $\mathsf{Q}^{-1}(1)$ *and* R_{Q} *its syntactic typed stamp. Then, a language is defined by a sentence of* $\Gamma_{\mathcal{Q},A}(\emptyset)$ *if and only if it is recognised by a restricted product of typed lp-substamps of* R_{Q} *over the alphabet* A*.*

As a consequence, we have that the *lp*-variety of languages $\mathcal{V}_{\mathcal{Q}}$ mapping the finite alphabet A to the closure under quotients of the Boolean algebra of languages definable in $\Gamma_{\mathcal{Q},A}(\emptyset)$ corresponds to the *lp*-pseudovariety of typed stamps that is generated by the elements of the form R_{Q}, with $\mathsf{Q} \in \mathcal{Q}$ (recall Theorem 9).

We illustrate Proposition 25 by instantiating Q with some of the most relevant examples of quantifiers appearing in the literature.

▶ **Example 26.** Let $\mathbb{B} = \{0,1\}$ be the two-element join semilattice. Then, $\mathsf{R}_\exists$ is given by the typed stamp $(\{0,1\}, \mu_\exists, \mathbb{B}, p_\exists, \{0,1\})$, where both $\mu_\exists$ restricted to $\{0,1\}$ and $p_\exists$ are the identity map between the respective underlying sets. On the other hand, for the quantifier $\exists^r_q$, one has $\mathsf{R}_{\exists^r_q} = (\{0,1\}, \mu_{\exists^r_q}, \mathbb{Z}/q\mathbb{Z}, p_{\exists^r_q}, \{0,1\})$, where $\mu_{\exists^r_q}(0) = [0]$, $\mu_{\exists^r_q}(1) = [1]$, and $p_{\exists^r_q}([k])$ is mapped to 1 if and only if $k - r$ is divisible by q. Finally, one may check that $\mathsf{R}_{\mathsf{Maj}}$ is the typed stamp $(\{0,1\}, \mu_{\mathsf{Maj}}, \mathbb{Z}, p_{\mathsf{Maj}}, \{0,1\})$ with $\mu_{\mathsf{Maj}}(0) = -1$, $\mu_{\mathsf{Maj}}(1) = 1$, and $p_{\mathsf{Maj}}(k) = 1$ if and only if $k > 0$.

Next, we illustrate how to define a typed stamp recognising a given numerical predicate. The intuitive idea is to use a product of monoids of the form $M_1 \times M_2 \times M_3$ where M_1 encodes the length of a word, M_2 encodes the first position marked by each variable, while M_3 encodes the last one. Let $(R : n \mapsto R_n)$ be a k-ary numerical predicate, and $\Upsilon = \{x_1, \dots, x_k\}$ a set of k variables. For each $\ell \in \mathbb{N}$ we define the map $\alpha_\ell : \mathbb{B} \to \mathbb{N}$ by setting $\alpha_\ell(0) = \ell$ and $\alpha_\ell(1) = 0$. We let $\mathbb{N} \circ \mathbb{B}$ and $\mathbb{N} \circ_r \mathbb{B}$ be, respectively the usual wreath and reversed wreath product of monoids, and N the submonoid of $\mathbb{N} \times (\mathbb{N} \circ \mathbb{B})^k \times (\mathbb{N} \circ_r \mathbb{B})^k$ generated by $\{1\} \times (\{\alpha_1\} \times \mathbb{B})^k \times (\{\alpha_1\} \times \mathbb{B})^k$. An easy computation shows that every element of N belongs to $\mathbb{N} \times (\{\alpha_\ell\}_{\ell \in \mathbb{N}} \times \mathbb{B})^k \times (\{\alpha_\ell\}_{\ell \in \mathbb{N}} \times \mathbb{B})^k$. For each subset $S \subseteq \Upsilon$ and each variable $x \in \Upsilon$ we set $n_{S,x} = (\alpha_1, 1)$ if $x \in S$ and $n_{S,x} = (\alpha_1, 0)$ otherwise. Finally, we let $\mathsf{S}_R = (A \times 2^\Upsilon, \nu_R, N, q_R, \{0,1\})$ be the typed stamp defined by

$$\begin{aligned} &\nu_R(a, S) = (1, (n_{S,x_i})_{i=1}^k, (n_{S,x_i})_{i=1}^k) \\ &q_R(n, (\alpha_{\ell_i}, \varepsilon_i)_{i=1}^k, (\alpha_{k_i}, \delta_i)_{i=1}^k) = 1 \iff \forall i \quad (\varepsilon_i = \delta_i = 1 \text{ and } k_i = n - \ell_i + 1) \\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{and } (\ell_1, \dots, \ell_k) \in R_n. \end{aligned}$$

▶ **Proposition 27.** *The typed stamp* S_R *recognises* L_R*, the language of* $(A \times 2^\Upsilon)^*$ *defined by* R*, via the subset* $\{1\}$*.*

We let $V_{\mathcal{N}}$ be the *lp*-pseudovariety generated by the typed stamps S_R, for $R \in \mathcal{N}$ an $|\Upsilon|$-ary numerical predicate.

Combining these constructions with Proposition 24, we have the following:

▶ **Proposition 28.** *Let* $\mathcal{Q}$ *be a set of quantifiers and* $\mathcal{N}$ *a set of numerical predicates such that every formula of* $\mathcal{Q}_A[\mathcal{N}]$ *admits a prenex normal form. Every language definable in* $\mathcal{Q}[\mathcal{N}]$ *is recognised by some typed stamp in*

$$\bigcup_{n \in \mathbb{N}} \underbrace{\mathrm{V}_{\mathcal{Q}} \square (\dots \square (\mathrm{V}_{\mathcal{Q}}}_{n \text{ times}} \square \mathrm{V}_{\mathcal{N}}) \dots). \tag{3}$$

On the other hand, as a consequence of the next result, we have that the elements of (3) do not recognise much more languages than the ones definable in $\mathcal{Q}[\mathcal{N}]$.

▶ **Proposition 29.** *Let* S *be the syntactic typed stamp of a formula* $\varphi \in \mathcal{Q}_A[\mathcal{N}](\Upsilon)$*. Then, every language recognised by a typed lp-substamp of* S *is definable in* $(\mathrm{FO} + \mathcal{Q})[\{=\} \cup \mathcal{N}]$*.*

Proof Sketch. We consider the case where $\Upsilon = \{x\}$ and $\varphi(x)$ has one free variable x. The general one is handled similarly. Let $\mathsf{S} = (A \times 2^{\{x\}}, \nu, N, q, Y)$ be the syntactic typed stamp of $\varphi(x)$, let B be a finite alphabet, and $h : B^* \to (A \times 2^{\{x\}})^*$ an *lp*-morphism. It suffices to show that the language $L = (q \circ \nu \circ h)^{-1}(1)$, is definable in $(\mathcal{Q} + FO)[\mathcal{N} \cup \{=\}]$. Take

$$\phi = \exists! z\ \varphi(z) \wedge (\vee_{h(b) \in A \times \{\{x\}\}} \mathsf{P}_b(z)).$$

Then, one may check that $L = L_\phi$. ◀

We finish this section with an example of application of Theorem 23.

▶ **Example 30.** Let $\mathcal{N}$ be a set of unary numerical predicates and $\mathcal{Q} = \{\exists\}$.

We write $\Sigma_{A \times 2^{\{x\}}}(V_\mathcal{N}) = (A \times 2^{\{x\}}, \nu, N, q, Y)$. By Example 26 we have that $V_\mathcal{Q}$ is the *lp*-pseudovariety generated by $\mathsf{R}_\exists$. Applying Theorem 23 and using a well-known fact about the structure of the space component of $\Sigma_Y(V_\mathcal{Q})$, we may derive that the BiM component of $\Sigma_A(V_\mathcal{Q} \,\square\, V_\mathcal{N})$ is the *Schützenberger product of* (N, q, Y) introduced in [10]. In the mentioned paper, the goal was to define a recogniser for the language $L_{\exists x\ \varphi(x)}$ given a recogniser for $L_{\varphi(x)}$, and the Schützenberger product of a BiM recognising $L_{\varphi(x)}$ played that role. We thus provide a different explanation for their result.

8 Discussion

In Proposition 14, we have shown that the transductions are bona fide duals of substitution in terms of Stone duality. This is crucial to the link between logic on words and topo-algebraic methods. Our main result, Theorem 23, allows one to compute the syntactic space of the Boolean algebra obtained by applying a very general form of quantifier to a variety of not necessarily regular languages. This result was first proved by Almeida and Weil in [1] for varieties of regular languages and quantifiers given by such. Several surprises have to be overcome for the generalisation: the closure properties of pseudovarieties of typed stamps are delicate, in the block-product of typed monoids, the 'right' finitely generated monoid must be identified. Finally, the B-monoid based on the semidirect product in Theorem 23 is *not* a BiM – even though the B-submonoid we seek is a BiM. More recently, the result of Theorem 23 in the special cases of the classical existential quantifier [10] and of quantifiers arising from finite semirings [11] has been obtained using codensity monads and transducers.

Boolean spaces are promising objects for separation results. As laid out in Section 7, the block product and substitution principle and their connection to semidirect products of B-monoids allow us, by induction, to compute the relevant spaces for important Boolean circuit complexity classes. The equations satisfied by these spaces may in turn help in finding a way of showing separations [9, 4].

Acknowledgements. The authors would like to thank Howard Straubing for fruitful discussions on the interaction of logic and algebraic language theory as well as the anonymous referees whose comments helped improve the final presentation of the paper.

References

1 Jorge Almeida and Pascal Weil. Free profinite semigroups over semidirect products. *Russian Mathematics*, 39(1):1–27, 1995.

2 David A. Mix Barrington, Kevin J. Compton, Howard Straubing, and Denis Thérien. Regular Languages in NC^1. *J. Comput. Syst. Sci.*, 44(3):478–499, 1992. doi:10.1016/0022-0000(92)90014-A.

3 Christoph Behle, Andreas Krebs, and Stephanie Reifferscheid. Typed Monoids – An Eilenberg-like Theorem for non regular Languages. *Electronic Colloq. on Comp. Complexity (ECCC)*, 18:35, 2011. URL: https://eccc.weizmann.ac.il/report/2011/035/.

4 Silke Czarnetzki and Andreas Krebs. Using duality in circuit complexity. In *Language and Automata Theory and Applications – 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, pages 283–294, 2016. doi:10.1007/978-3-319-30000-9_22.

5 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 260–270, 1981. doi:10.1109/SFCS.1981.35.

6 Mai Gehrke. Duality in computer science. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'16, New York, NY, USA, July 5-8, 2016*, pages 12–26, 2016. doi:10.1145/2933575.2934575.

7 Mai Gehrke, Serge Grigorieff, and Jean-Eric Pin. A topological approach to recognition. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, pages 151–162, 2010. doi:10.1007/978-3-642-14162-1_13.

8 Mai Gehrke and Andreas Krebs. Stone duality for languages and complexity. *SIGLOG News*, 4(2):29–53, 2017. doi:10.1145/3090064.3090068.

9 Mai Gehrke, Andreas Krebs, and Jean-Éric Pin. Ultrafilters on words for a fragment of logic. *Theor. Comput. Sci.*, 610:37–58, 2016. doi:10.1016/j.tcs.2015.08.007.

10 Mai Gehrke, Daniela Petrisan, and Luca Reggio. The Schützenberger Product for Syntactic Spaces. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 112:1–112:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.112.

11 Mai Gehrke, Daniela Petrişan, and Luca Reggio. Quantifiers on languages and codensity monads. To appear in LICS, 2017. URL: https://arxiv.org/abs/1702.08841.

12 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999. doi:10.1007/978-1-4612-0539-5.

13 Peter T. Johnstone. *Stone spaces*, volume 3 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1986. Reprint of the 1982 edition.

14 Andreas Krebs, Klaus-Jörn Lange, and Stephanie Reifferscheid. Characterizing TC^0 in Terms of Infinite Groups. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 496–507, 2005. doi:10.1007/978-3-540-31856-9_41.

15 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, Basel, Switzerland, 1994.

16 Howard Straubing. On logical descriptions of regular languages. In *LATIN 2002: Theoretical Informatics, 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002, Proceedings*, pages 528–538, 2002. doi:10.1007/3-540-45995-2_46.

17 Pascal Tesson and Denis Thérien. Logic meets algebra: the case of regular languages. *Logical Methods in Computer Science*, 3(1), 2007. doi:10.2168/LMCS-3(1:4)2007.

18 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.

A Projective and direct limits

A *projective system* (also known as an inverse limit system, or a cofiltered diagram) $\mathcal{F}$ of sets assigns to each element i of a *directed* partially ordered set I, a set S_i, and to each ordered pair $i \geq j$ in I, a map $f_{i,j} \colon S_i \to S_j$ so that for all $i, j, k \in I$ with $i \geq j \geq k$ we have $f_{i,i} = id_{S_i}$ and $f_{j,k} \circ f_{i,j} = f_{i,k}$. The *projective limit* (or inverse limit or cofiltered limit) of $\mathcal{F}$, denoted $\varprojlim \mathcal{F}$, comes equipped with projection maps $\pi_i : \varprojlim \mathcal{F} \to S_i$ compatible with the system. That is, for $i, j \in I$ with $i \geq j$, $f_{i,j} \circ \pi_i = \pi_j$. Further, it satisfies the following *universal property*: whenever $\{\pi'_i : S' \to S_i\}_{i \in I}$ is a family of maps satisfying $f_{i,j} \circ \pi'_i = \pi'_j$ for all $i \geq j$, there exists a map $g : S' \to \varprojlim \mathcal{F}$ satisfying $\pi'_i = \pi_i \circ g$, for all $i \in I$.

There are several things worth noting about this notion. First, the projective limit of $\mathcal{F}$ may be constructed as follows:

$$\varprojlim \mathcal{F} = \left\{ (s_i)_{i \in I} \in \prod_{i \in I} S_i \mid f_{i,j}(s_i) = s_j \text{ whenever } i \geq j \right\}.$$

Second, projective limits of finite sets, called *profinite sets*, are equivalent to Boolean spaces. If each S_i is finite, then it is a Boolean space in the discrete topology, and the projective limit is a closed subspace of the product and thus again a Boolean space. Conversely, a Boolean space is the projective limit of the projective system of its finite continuous quotients.

Third, one also has projective systems and projective limits of richer structures than sets, such as algebras, topological spaces, maps between sets, etc. In these enriched settings the connecting maps are then required to be morphisms of the appropriate kind. A very useful fact, which is used throughout this work, is that in all these settings, the projective limits are given as for sets with the obvious enriched structure.

The notion dual to projective limit, obtained by reversing the directions of the maps, is that of *direct limit* (also known as an injective limit or inductive limit or filtered colimit), and it is denoted $\varinjlim$. It corresponds to the construction invoked in Definition 12.

B Appendix to Section 3

B.1 Proof of Proposition 7

We write $\mathsf{R} = (A, \mu, M, p, X)$ and we let $\mathsf{R}_{\mathcal{B}} = (A, \mu_{\mathcal{B}}, M_{\mathcal{B}}, p_{\mathcal{B}}, X_{\mathcal{B}})$ be the syntactic B-stamp of $\mathcal{B}$. It is clear that if $\mathsf{R}_{\mathcal{B}}$ factors through R, then every language recognised by $\mathsf{R}_{\mathcal{B}}$ is also recognised by R and, in particular, $\mathcal{B}$ is recognised by R. Conversely, suppose that $\mathcal{B}$ is recognised by R. Then, the kernel of μ is contained in the syntactic congruence $\sim_{\mathcal{B}}$, and therefore, the syntactic morphism $\mu_{\mathcal{B}}$ factors through μ, say $g \circ \mu = \mu_{\mathcal{B}}$. On the other hand, the dual of the embedding $\mathcal{B} \hookrightarrow \mathsf{Clopen}(X)$ yields a continuous quotient map φ form X to $X_{\mathcal{B}}$. Is is easy to check that the triple (id, g, φ) indeed defined a morphism from R to $\mathsf{R}_{\mathcal{B}}$. ◀

C Appendix to Section 4

C.1 Proof of Proposition 13

For a given map $\zeta : A \to B$ between finite alphabets, and a set $\Upsilon = \{x_1, \ldots, x_k\}$ of variables, we define the map $\zeta \otimes \Upsilon : A^* \otimes \Upsilon \to B^* \otimes \Upsilon$ by $(\zeta \otimes \Upsilon)(w_{x_1 = i_1, \ldots, x_k = i_k}) = \zeta^*(w)_{x_1 = i_1, \ldots, x_k = i_k}$. Recall that, by Property (LC.3) of a logic class, ζ induces a map $\zeta_{\Gamma, \Upsilon} : \Gamma_B(\Upsilon) \to \Gamma_A(\Upsilon)$. We first observe the following:

▶ **Lemma 31.** *Let Γ be a logic class, Υ a finite set of variables, and $\zeta : A \to B$ a map between finite alphabets. Then, for every formula $\varphi \in \Gamma_B(\Upsilon)$ and Υ-structure $u \in A^* \otimes \Upsilon$, we have*

$$u \models \zeta_{\Gamma,\Upsilon}(\varphi) \iff (\zeta \otimes \Upsilon)(u) \models \varphi.$$

The proof is omitted as it follows straightforward from induction on the construction of a formula.

To prove Proposition 13 we shall also argue by induction on the construction of a formula. This requires a slight extension of the substitution map defined in Section 4.1.

▶ **Definition 32.** Let Γ and Λ be logic classes, Υ a finite set of variables, and x a variable that does not belong to Υ. Given a finite Boolean subalgebra $\mathcal{B} \subseteq \Lambda_A(\Upsilon \cup \{x\})$ and a finite set of variables Ψ disjoint from $\Upsilon \cup \{x\}$, the map $\sigma_{\mathcal{B},\Psi} : \Gamma_{C_\mathcal{B}}(\Psi) \to \mathcal{Q}_A[\mathcal{N}](\Psi \cup \Upsilon \cup \{x\})$ is the natural extension of the substitution map $\sigma_\mathcal{B}$.

Proof of Proposition 13. Properties (LC.1) and (LC.2) follow immediately from the fact that Γ and Λ are logic classes and from the definition of $\Gamma \circ \Lambda$. To prove (LC.3), we fix a map $\zeta : A \to B$ and, in order to simplify, we take $\Upsilon = \emptyset$. No additional difficulty arises from the general case. By definition of $\Gamma \circ \Lambda$, it suffices to show that, for every finite Boolean subalgebra $\mathcal{B} \subseteq \Lambda_B(x)$, the image of $\zeta_{\Gamma\circ\Lambda,\emptyset} \circ \sigma_\mathcal{B}$ is contained in $(\Gamma \circ \Lambda)_A(\emptyset)$. Since Λ is a logic class, the function ζ defines a morphism $\zeta_{\Lambda,\{x\}} : \Lambda_B(x) \to \Lambda_A(x)$. Let $\mathcal{B}'$ be the Boolean algebra generated by $\zeta_{\Lambda,\{x\}}(\mathcal{B})$. We claim that the set of atoms of $\mathcal{B}'$ is given by $\zeta_{\Lambda,\{x\}}(\mathsf{At}(\mathcal{B}))$. Indeed, this follows from the equivalence

$$w_{x=i} \models \zeta_{\Lambda,\{x\}}(\varphi) \iff \zeta(w)_{x=i} \models \varphi, \text{ for every } \varphi \in \mathcal{B} \text{ and } w \in A^*,$$

which is given by Lemma 31. Hence, the bijection $\zeta' : C_\mathcal{B} \to C_{\mathcal{B}'}$ induced by $\zeta_{\Lambda,\{x\}}$ defines a map $\zeta'_{\Gamma,\emptyset} : \Gamma_{C_\mathcal{B}}(\emptyset) \to \Gamma_{C_{\mathcal{B}'}}(\emptyset)$, and one may check the equality $\sigma_{\mathcal{B}'} \circ \zeta'_{\Gamma,\emptyset} = \zeta_{\Gamma\circ\Lambda,} \circ \sigma_\mathcal{B}$. This completes the proof. ◀

C.2 Proof of Proposition 14

Let $w \in A^*$, $u \in C_\mathcal{B}^*$, and $c \in C_\mathcal{B}$, and denote by $\varphi_c(x)$ the atom of $\mathcal{B}$ that the letter c stands for. Then the fact that ξ is dual to the inclusion means

$$w_{x=i} \vDash \varphi_c(x) \iff c = \xi(w_{x=i})$$

and by the definition of letter predicates we have

$$u_{x=i} \vDash \mathsf{P}_c(x) \iff u_i = c.$$

So, by definition of $\tau_\mathcal{B}$, for each $i \in \{1, \ldots, |w|\}$ and each $c \in C_\mathcal{B}$, we have

$$w_{x=i} \vDash \varphi_c(x) \iff (\tau_\mathcal{B}(w))_{x=i} \vDash \mathsf{P}_c(x).$$

Since the validity of quantifiers is determined by the truth values of the formulas in their scope at all points of the model, and since $\psi \in \Gamma_{C_\mathcal{B}}(\emptyset)$ and $\tau_\mathcal{B}(\psi)$ are built up identically once the substitutions of $\mathsf{P}_c(x)$ by $\varphi_c(x)$ have been made, it follows that, for each $\psi \in \Gamma_{C_\mathcal{B}}(\emptyset)$, we have

$$\tau_\mathcal{B}(w) \in L_\psi \iff w \in L_{\sigma_\mathcal{B}(\psi)}. \tag{4}$$

However

$$w \in L_{\sigma_{\mathcal{B}}(\psi)} \iff L_{\sigma_{\mathcal{B}}(\psi)} \in p(w) \iff L_\psi \in \Sigma_{\mathcal{B}}(p(w))$$

so that

$$\tau_{\mathcal{B}}(w) \in L_\psi \iff L_\psi \in \Sigma_{\mathcal{B}}(p(w))$$

and thus $\Sigma_{\mathcal{B}}(p(w)) = q(\tau_{\mathcal{B}}(w))$ as required. ◀

D Appendix to Section 5

D.1 Proof of Proposition 20

It is enough to prove the claim for languages of the form

$$L = ((p \,\square\, q) \circ (\mu \,\square_q\, \nu))^{-1}(x, y),$$

for some $(x, y) \in X \times Y$. Note that every other language recognised by $\mathsf{R} \,\square\, \mathsf{S}$ is a Boolean combination of languages of this form. Let $w \in A^*$ be a word. By the definitions, w belongs to L if and only if

$$(p \circ \mu \circ \tau_{\mathsf{S}}(w), q \circ \nu(w)) = (x, y).$$

But this means that $L = \tau_{\mathsf{S}}^{-1}(K_1) \cap K_2$ where $K_1 = (p \circ \mu)^{-1}(x)$ and $K_2 = (q \circ \nu)^{-1}(y)$ are languages recognised, respectively, by R and by the lp-substamp of S determined by the injective map $A^* \hookrightarrow (A \times 2^{\{x\}})^*$ sending a to $(a, \emptyset)$. ◀

E Appendix to Section 6

E.1 Proof of Proposition 22

In Section 3.4 we defined the B-stamp $\Sigma_Y(V)$ to be the projective limit of a certain family of BiM presentations. We use the notation of that section in the rest of the proof, which we briefly recall here. If I is the set indexing the finite continuous quotients of Y, which are denoted $\pi_i : Y \twoheadrightarrow Y_i$, we say that $i \geq j$ provided there exists a quotient map $h_{i,j} : Y_i \twoheadrightarrow Y_j$ such that $h_{i,j} \circ \pi_i = \pi_j$. In particular, writing $\Sigma_{Y_i}(V) = (Y_i, \mu_i, M_i, p_i, X_i)$, we have a connecting morphism $\Phi_{i,j} = (h_{i,j}, g_{i,j}, \varphi_{i,j})$ whenever $i \geq j$.

We start by defining the left action of N on X. Given $n \in N$, we let $\lambda_n : Y \to Y$ be the left action of N on Y (recall Definition 2). For each $i \in I$, the co-restriction to the image of the map $\pi_i \circ \lambda_n$ is a continuous quotient of Y. Thus, there exists an index $\varepsilon(i) \in I$ such that $\pi_{\varepsilon(i)}$ is the co-restriction of $\pi_i \circ \lambda_n$ to $Y_{\varepsilon(i)}$. In order to make more explicit at each step the set to which a certain element belongs, we denote by $h_i : Y_{\varepsilon(i)} \hookrightarrow Y_i$ the inclusion map. In particular, the equality $h_i \circ \pi_{\varepsilon(i)} = \pi_i \circ \lambda_n$ holds. Since V is a $\mathcal{C}$-pseudovariety, the map h_i yields a representation of $\Sigma_{Y_{\varepsilon(i)}}(V)$ as a B-substamp of $\Sigma_{Y_i}(V)$. We denote by $g_i : M_{\varepsilon(i)} \hookrightarrow M_i$ and $\varphi_i : X_{\varepsilon(i)} \hookrightarrow X_i$ the corresponding inclusion maps. In particular, we have the equality

$$\varphi_i \circ p_{\varepsilon(i)} \circ \mu_{\varepsilon(i)} = p_i \circ \mu_i \circ h_i^*. \tag{5}$$

On the other hand, whenever $i \geq j$, it is an easy observation that the codomain of the map $h_{i,j} \circ h_i$ is precisely $Y_{\varepsilon(j)}$. Therefore, we also have $\varepsilon(i) \geq \varepsilon(j)$. Note that, for $\kappa \in \{h, g, \varphi\}$, we have

$$\kappa_j \circ \kappa_{\varepsilon(i),\varepsilon(j)} = \kappa_{i,j} \circ \kappa_i. \tag{6}$$

To define the map $\psi_{\ell,n}$, we remark that X is the subspace of $\prod_{i\in I} X_i$ that consists of the tuples $(x_i)_{i\in I}$ satisfying $\varphi_{i,j}(x_i) = x_j$ for every $i \geq j$. Then, we set

$$\psi_{\ell,n}((x_i)_{i\in I}) = (\varphi_i(x_{\varepsilon(i)}))_{i\in I}. \tag{7}$$

This is a well-defined continuous map provided $\varphi_j(x_{\varepsilon(j)}) = \varphi_{i,j} \circ \varphi_i(x_{\varepsilon(i)})$, whenever $i \geq j$. But this follows immediately from the definition of X, together with (6) for $\kappa = \varphi$. More generally, we define $\psi_{r,n}$.

Finally, it is easy to see that setting $n \cdot x = \psi_{\ell,n}(x)$ and $x \cdot n = \psi_{r,n}(x)$, for $n \in N$ and $x \in X$, defines a biaction.

Now, given $(y_1, \ldots, y_k) \in Y^*$, we may compute:

$$\begin{aligned}
\psi_{\ell,n} \circ p \circ \mu(y_1, \ldots, y_k) &= \psi_{\ell,n}[(p_i \circ \mu_i \circ \pi_i^*(y_1, \ldots, y_k))_{i\in I}] && \text{by definition of } X\\
&= (\varphi_i \circ p_{\varepsilon(i)} \circ \mu_{\varepsilon(i)} \circ \pi^*_{\varepsilon(i)}(y_1, \ldots, y_k))_{i\in I} && \text{by (7)}\\
&= (p_i \circ \mu_i \circ h_i^* \circ \pi^*_{\varepsilon(i)}(y_1, \ldots, y_k))_{i\in I} && \text{by (5)}\\
&= (p_i \circ \mu_i \circ \pi_i^* \circ \lambda_n^*(y_1, \ldots, y_k))_{i\in I} && \text{because } \pi_i \circ \lambda_n = h_i \circ \pi_{\varepsilon(i)}.
\end{aligned}$$

This computation proves (1). Equality (2) is derived similarly. ◀

F Appendix to Section 7

F.1 Proof of Proposition 25

It suffices to consider the case where we are given a sentence of the form

$$\varphi = (\mathsf{Q}x \bigvee_{a\in B} \mathsf{P}_a(x)),$$

for a certain subset $B \subseteq A$. Let $h : A^* \to \{0,1\}^*$ be the unique homomorphism that sends the letter a to 1 if and only if a belongs to B. Then, by the way quantifiers are interpreted, we have that the models of φ are precisely the elements of $(\mathsf{Q} \circ h)^{-1}(1)$ and thus, L_φ is recognised by the typed lp-substamp of R_Q defined by h. ◀

F.2 Proof of Proposition 27

We illustrate the proof in the case R is a unary predicate. We first observe that, for every $\ell, \ell_1, \ell_2 \in \mathbb{N}$, the equalities

$$\alpha_\ell \cdot 0 = \alpha_\ell = 0 \cdot \alpha_\ell; \quad \alpha_\ell \cdot 1 = \alpha_0 = 1 \cdot \alpha_\ell; \quad \textit{and} \quad \alpha_{\ell_1} + \alpha_{\ell_2} = \alpha_{\ell_1+\ell_2};$$

hold. Given a word $u = (a_1, S_1) \cdots (a_m, S_m) \in (A \times 2^{\{x\}})^*$ with at least one marked position, we set

$$i_{\min} = \min\{j \mid S_j \neq \emptyset\} \quad \text{and} \quad i_{\max} = \max\{j \mid S_j \neq \emptyset\}.$$

Then, one may compute

$$\nu_R(u) = (m, (\alpha_{i_{\min}}, 1), (\alpha_{m-i_{\max}+1}, 1))$$

Thus, u belongs to $A^* \otimes \{x\}$ if and only if $i_{\min} = m - i_{\max} + 1$, and in that case the unique marked position of w is $i_{\min}$. This means that u is a model of R if and only if $p_R \circ \nu_R(u) = 1$. On the other hand, if w is a word over A, then we have

$$\nu_R(w) = (\alpha_\ell, 0),$$

for some $\ell \in \mathbb{N}$. Thus, $p_R \circ \nu_R(w) = 0$. This proves that $L_R = (p_R \circ \nu_R)^{-1}(1)$ as intended. ◀

A Decidable Intuitionistic Temporal Logic*

Joseph Boudou[1], Martín Diéguez[2], and David Fernández-Duque[3]

1 **Institut de Recherche en Informatique de Toulouse, Toulouse University, Toulouse, France**
2 **Institut de Recherche en Informatique de Toulouse, Toulouse University, Toulouse, France**
3 **Institut de Recherche en Informatique de Toulouse, Toulouse University, Toulouse, France**

Abstract

We introduce the logic $\mathsf{ITL}^{\mathsf{e}}$, an intuitionistic temporal logic based on structures $(W, \preccurlyeq, S)$, where $\preccurlyeq$ is used to interpret intuitionistic implication and S is a $\preccurlyeq$-monotone function used to interpret temporal modalities. Our main result is that the satisfiability and validity problems for $\mathsf{ITL}^{\mathsf{e}}$ are decidable. We prove this by showing that the logic enjoys the strong finite model property. In contrast, we also consider a 'persistent' version of the logic, $\mathsf{ITL}^{\mathsf{p}}$, whose models are similar to Cartesian products. We prove that, unlike $\mathsf{ITL}^{\mathsf{e}}$, $\mathsf{ITL}^{\mathsf{p}}$ does not have the finite model property.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases intuitionistic logic, temporal logic, products of modal logics

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.14

1 Introduction

Intuitionistic logic [6, 22] and its modal extensions [9, 27, 28] play a crucial role in the area of computer science and artificial intelligence. For instance, Pearce's Equilibrium Logic [26], which characterises the Answer Set semantics [21, 23] of logic programs (ASP), is defined in terms of the intermediate logic of Here and There [15], together with a minimisation criterion. Extensions of Here and There logic allowed the ASP paradigm, already used in a wide range of domains [1, 3, 14, 16, 25], to be applied to reasoning about temporal or epistemic scenarios [5, 10] while satisfying the theorem of strong equivalence [4, 20, 10], central to logic programming and nonmonotonic reasoning.

Such modal extensions of Here and There logic are simple cases of a *modal intuitionistic logic;* in general, the study of such logics can be a challenging enterprise [28]. In particular, there is a huge gap that must be filled regarding combinations of intuitionistic and linear-time temporal logic. Nevertheless, there have been several efforts in this direction, including logics with 'past' and 'future' tenses [9] or with 'next' $\bigcirc$, 'eventually' $\Diamond$ and/or 'henceforth' $\Box$ modalities. The main contributions to the field include the following:

- Davies' intuitionistic temporal logic with $\bigcirc$ [7] was provided Kripke semantics and a complete deductive system by Kojima and Igarashi [18].
- Logics with $\bigcirc, \Box$ were axiomatized by Kamide and Wansing [17], where $\Box$ was interpreted over bounded time.
- Nishimura [24] provided a sound and complete axiomatization for an intuitionistic variant of the propositional dynamic logic PDL.

* This research was partially supported by ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02.

- Balbiani and Diéguez [2] axiomatized the Here and There variant of LTL with $\bigcirc, \Diamond, \Box$.
- Davoren [8] introduced topological semantics for temporal logics and Fernández-Duque [11] proved the decidability of a logic with $\bigcirc, \Diamond$ and a universal modality based on topological semantics.

With the exception of [8, 11], semantics for intuitionistic LTL use frames of the form $(W, \preccurlyeq, S)$, where $\preccurlyeq$ is a partial order used to interpret the intuitionistic implication and S is a binary relation used to interpret temporal operators. Since we are interested in linear time, we will restrict our attention to the case where S is a function. Thus, for example, $\bigcirc p$ is true on some world $w \in W$ whenever p is true on $S(w)$. Note, however, that S cannot be an arbitrary function. Intuitionistic semantics have the feature that, for any formula φ and worlds $w \preccurlyeq v \in W$, if φ is true on w then it must also be true on v; that is, truth is *monotone*. If we want this property to be preserved by formulas involving $\bigcirc$, we need for $\preccurlyeq$ and S to satisfy certain confluence properties. In the literature, one generally considers frames satisfying

1. $w \preccurlyeq v$ implies $S(w) \preccurlyeq S(v)$ (*forward confluence,* or simply *confluence*), and
2. if $u \succcurlyeq S(w)$, there is $v \succcurlyeq w$ such that $S(v) = u$ (*backward confluence*).

We will call frames satisfying these conditions *persistent frames* (see Sec. 3), mainly due to the fact that they are closely related to (persistent) products of modal logics [12]. Persistent frames for intuitionistic LTL are the frames of the modal logic S4 × LTL, which is non-axiomatizable. For this reason, it may not be surprising that it is unknown whether the intuitionistic temporal logic of persistent frames, which we denote $\mathsf{ITL}^{\mathsf{p}}$, is decidable.

However, as we will see in Proposition 1, only forward confluence is needed for truth of all formulas to be monotone, even in the presence of $\Diamond$ and $\Box$. The frames satisfying this condition are, instead, related to *expanding* products of modal logics [13], which are often decidable even when the corresponding product is non-axiomatizable. This suggests that dropping the backwards confluence could also lead to a more manageable intuitionistic temporal logic. This logic, which we denote $\mathsf{ITL}^{\mathsf{e}}$, is the focus of the present paper and, as we will prove in this paper, it enjoys a crucial advantage over $\mathsf{ITL}^{\mathsf{p}}$: $\mathsf{ITL}^{\mathsf{e}}$ has the strong finite model property[1] (hence it is decidable), but $\mathsf{ITL}^{\mathsf{p}}$ does not. In fact, to the best of our knowledge, $\mathsf{ITL}^{\mathsf{e}}$ is the first known decidable intuitionistic temporal logic that

1. is conservative over propositional intuitionistic logic,
2. includes (or can define) the three modalities $\bigcirc, \Diamond, \Box$, and
3. is interpreted over infinite time.

2 Syntax and semantics

We will work in the language $\mathcal{L}$ of LTL given by the following grammar:

$$\varphi, \psi \coloneqq p \mid \bot \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi,$$

where p is an element of a countable set of propositional variables $\mathbb{P}$. Given any formula φ, we write $\mathrm{SF}(\varphi)$ for the set of subformulas of φ and $|\varphi|$ for the cardinality of $\mathrm{SF}(\varphi)$.

A *dynamic poset* is a tuple $(W, \preccurlyeq, S)$, where W is a non-empty set of states, $\preccurlyeq$ is a partial order, and S is a function from W to W that satisfies the following *(forward) confluence* condition:

$$\text{for all } w, v \in W, \text{ if } w \preccurlyeq v \text{ then } S(w) \preccurlyeq S(v). \tag{1}$$

[1] That is, there is a computable function bounding the size of the smallest model, if there is one.

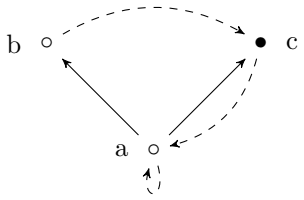


Figure 1 Example of an ITL$^{\mathsf{e}}$ model $\mathcal{M} = (W, \preccurlyeq, S, V)$, where $\preccurlyeq$ is the reflexive closure of the (already transitive) relation indicated by the solid arrows, S is the relation indicated by the dashed arrows, and a black dot indicates that the variable p is true, so that we only have $p \in V(c)$. Then, the reader may readily verify that $\mathcal{M}, b \vDash \bigcirc p$ but $\mathcal{M}, b \nvDash p$, while $\mathcal{M}, c \vDash p$ but $\mathcal{M}, c \nvDash \bigcirc p$. From this it follows that $\mathcal{M}, a \nvDash (\bigcirc p \to p) \vee (p \to \bigcirc p)$.

An *intuitionistic dynamic model,* or simply a *model,* is a tuple $\mathcal{M} = (W, \preccurlyeq, S, V)$ consisting of a dynamic poset equipped with a valuation function V from W to sets of propositional variables satifying the *monotonicity* condition:

$$\text{for all } w, v \in W, \text{ if } w \preccurlyeq v \text{ then } V(w) \subseteq V(v). \tag{2}$$

In the standard way, we define $S^0(w) = w$ and, for all $k > 0$, $S^k(w) = S\left(S^{k-1}(w)\right)$. Then we define the satisfaction relation $\vDash$ inductively by:

$$\begin{array}{ll}
\mathcal{M}, w \vDash p & \text{iff } p \in V(w) \\
\mathcal{M}, w \vDash \bigcirc\varphi & \text{iff } \mathcal{M}, S(w) \vDash \varphi \\
\mathcal{M}, w \vDash \bot & \text{never} \\
\mathcal{M}, w \vDash \Diamond\varphi & \text{iff } \exists k \text{ s.t. } \mathcal{M}, S^k(w) \vDash \varphi \\
\mathcal{M}, w \vDash \varphi \wedge \psi & \text{iff } \mathcal{M}, w \vDash \varphi \text{ and } \mathcal{M}, w \vDash \psi \\
\mathcal{M}, w \vDash \Box\varphi & \text{iff } \forall k,\ \mathcal{M}, S^k(w) \vDash \varphi \\
\mathcal{M}, w \vDash \varphi \vee \psi & \text{iff } \mathcal{M}, w \vDash \varphi \text{ or } \mathcal{M}, w \vDash \psi \\
\mathcal{M}, w \vDash \varphi \to \psi & \text{iff } \forall v \succcurlyeq w,\ \text{if } \mathcal{M}, v \vDash \varphi \text{ then } \mathcal{M}, v \vDash \psi
\end{array}$$

Given a model $\mathcal{M} = (W, \preccurlyeq, S, V)$, a set Σ of formulas, and $w \in W$, we write $\Sigma_{\mathcal{M}}(w)$ for the set $\{\psi \in \Sigma \mid \mathcal{M}, w \vDash \psi\}$; the subscript '$\mathcal{M}$' is omitted when it is clear from the context. An *eventuality* in $\mathcal{M}$ is a pair (w, φ), where $w \in W$ and φ is a formula such that either $\varphi = \Diamond\psi$ for some formula ψ and $\mathcal{M}, w \vDash \varphi$, or $\varphi = \Box\psi$ for some formula ψ and $\mathcal{M}, w \nvDash \varphi$. The *fulfillment* of an eventuality (w, φ) is the finite sequence $v_0 \dots v_n$ of states of the model such that

1. for all $k \leq n$, $v_0 = S^k(w)$,
2. if $\varphi = \Diamond\psi$ then $\mathcal{M}, v_n \vDash \psi$ and for all $k < n$, $\mathcal{M}, v_k \nvDash \psi$, and
3. if $\varphi = \Box\psi$ then $\mathcal{M}, v_n \nvDash \psi$ and for all $k < n$, $\mathcal{M}, v_k \vDash \psi$.

A formula φ is *satisfiable over a class* Ω *of models* if there is a model $\mathcal{M} \in \Omega$ and a world w so that $\mathcal{M}, w \vDash \varphi$, and *valid over* Ω if, for every world w of every model $\mathcal{M} \in \Omega$, $\mathcal{M}, w \vDash \varphi$. Satisfiability (resp. validity) over the class of all intuitionisitic dynamic models is called *satisfiability* (resp. *validity*) for the *expanding domain intuitionisitic temporal logic* ITL$^{\mathsf{e}}$. We will justify this terminology in the next section. First, we remark that dynamic posets impose the minimal conditions on S and $\preccurlyeq$ in order to preserve the upwards-closure of valuations of formulas. Below, we will use the notation $[\![\varphi]\!] = \{w \in W \mid \mathcal{M}, w \vDash \varphi\}$.

▶ **Proposition 1.** *Let $\mathcal{D} = (W, \preccurlyeq, S)$, where $(W, \preccurlyeq)$ is a poset and $S\colon W \to W$ is any function. Then, the following are equivalent:*
1. *S satisfies the confluence property (1);*
2. *for every valuation V on W and every formula φ, $[\![\varphi]\!]$ is upwards-closed under $\preccurlyeq$.*

Proof. That 1 implies 2 follows by a standard structural induction on φ. The case where $\varphi \in \mathbb{P}$ follows from the condition on V and most inductive steps are routine. Consider the case where $\varphi = \Box\psi$, and suppose that $w \preccurlyeq v$ and $w \in [\![\varphi]\!]$. Then, for all $i \in \mathbb{N}$, $\mathcal{M}, S^i(w) \vDash \psi$. Since S is confluent, an easy induction shows that, for all $i \in \mathbb{N}$, $S^i(w) \preccurlyeq S^i(v)$. Therefore, from the induction hypothesis we obtain that $\mathcal{M}, S^i(v) \vDash \psi$ for all i, hence $v \in [\![\varphi]\!]$. Other cases are similar or easier.

Now we prove that 2 implies 1 by contrapositive. Suppose that $(W, \preccurlyeq, S)$ does *not* satisfy (1), so that there are $w \preccurlyeq v$ such that $S(w) \not\preccurlyeq S(v)$. Choose $p \in \mathbb{P}$ and define $V(u) = \{p\}$ if $S(w) \preccurlyeq u$, $V(u) = \varnothing$ otherwise. It is easy to see that V satisfies the monotonicity condition (2). But, $p \notin V(S(v))$, from which it follows that $(\mathcal{D}, V), w \vDash \bigcirc p$ but $(\mathcal{D}, V), v \nvDash \bigcirc p$. ◀

We are concerned with the satisfiability and validity problems for $\mathsf{ITL}^{\mathsf{e}}$. Observe that satisfiability in propositional intuitionistic logic is equivalent to satisfiability in classical propositional logic. This is because, if φ is classically satisfiable, it is trivially intuitionistically satisfiable in a one-world model; conversely, if φ is intuitionistically satisfiable, it is satisfiable in a finite model, hence in a maximal world of that finite model, and the generated submodel of a maximal world is a classical model. Thus it may be surprising that the same is not the case for intuitionistic temporal logic:

▶ **Lemma 2.** *Any formula φ of the temporal language that is classically satisfiable is satisfiable in a dynamic poset. However, there is a formula satisfiable on a dynamic poset that is not classically satisfiable.*

Proof. If φ is satisfied on a classical LTL model $\mathcal{M}$, then we may regard $\mathcal{M}$ as an intuitionistic model by letting $\preccurlyeq$ be the identity. On the other hand, consider the formula $\neg\bigcirc p \wedge \neg\bigcirc\neg p$ (recall that $\neg\theta$ is a shorthand for $\theta \to \bot$). Classically, this formula is equivalent to $\neg\bigcirc p \wedge \bigcirc p$, and hence unsatisfiable. Define a model $\mathcal{M} = (W, \preccurlyeq, S, V)$, where $W = \{w, v, u\}$, $x \preccurlyeq y$ if $x = y$ or $x = v$, $y = u$, $S(w) = v$ and $S(x) = x$ otherwise, $V(u) = \{p\}$ and $V(v) = V(w) = \varnothing$. Then, one can check that $\mathcal{M}, w \vDash \neg\bigcirc p \wedge \neg\bigcirc\neg p$. ◀

Hence the decidability of the intuitionistic satisfiability problem is not a corollary of the classical case. In Section 5, we will prove that both the satisfiability and the validity problems are decidable.

3 Expanding and persistent frames

In this section, we discuss expanding and persistent models, and compare them to dynamic models as we have defined above.

3.1 Expanding model property

The logic $\mathsf{ITL}^{\mathsf{e}}$ is closely related to *expanding products* of modal logics [13]. In this subsection, we introduce stratified and expanding frames, and show that satisfiability and validity on arbitrary models is equivalent to satisfiability and validity on expanding models. To do this, it is convenient to represent posets using acyclic graphs.

▶ **Definition 3.** A *directed acyclic graph* is a tuple $(W, \uparrow)$, where W is a set of vertices, $\uparrow \subseteq W \times W$ is a set of edges whose reflexive, transitive closure $\uparrow^*$ is antisymmetric. We will tacitly identify $(W, \uparrow)$ with the poset $(W, \uparrow^*)$. A *path* from w_1 to w_2 is a finite sequence $v_0 \dots v_n \in W$ such that $v_0 = w_1$, $v_n = w_2$ and for all $k < n$, $v_k \uparrow v_{k+1}$. A *tree* is an acyclic graph $(W, \uparrow)$ with an element $r \in W$, called the root, such that for all $w \in W$ there is a unique path from r to w. A poset $(W, \preccurlyeq)$ is also a tree if there is a relation $\uparrow$ on $W \times W$ such that $(W, \uparrow)$ is a tree and $\preccurlyeq = \uparrow^*$.

▶ **Definition 4.** A model $\mathcal{M} = (W, \preccurlyeq, S, V)$ is *stratified* if there is a partition $\{W_n\}_{n<\omega}$ of W such that

1. each W_n is closed under $\preccurlyeq$,
2. for all n, there is relation $\uparrow_n$ such that $(W_n, \preccurlyeq\restriction_{W_n})$ is a tree, and
3. if $w \in W_n$ then $S(w) \in W_{n+1}$.

If $\mathcal{M}$ is stratified, we write $\preccurlyeq_n, S_n$, and V_n instead of $\preccurlyeq\restriction_{W_n}, S\restriction_{W_n}$, and $V\restriction_{W_n}$ and write $\mathcal{M}_n = (W_n, \preccurlyeq_n, V_n)$. If moreover we have that $S(w) \preccurlyeq S(v)$ implies $w \preccurlyeq v$, then we say that $\mathcal{M}$ is an *expanding model.*

Given a finite, non-empty set of formulas Σ closed under subformulas, a model $\mathcal{M} = (W, \preccurlyeq, S, V)$, and a state $w \in W$, we will construct a stratified model $\mathcal{M}^{\mathrm{e}} = (W^{\mathrm{e}}, \preccurlyeq^{\mathrm{e}}, S^{\mathrm{e}}, V^{\mathrm{e}})$ such that for the root w^{e} of W^{e}_0, $\Sigma(w^{\mathrm{e}}) = \Sigma(w)$. To this end, we first define the set $\mathcal{D} = \mathbb{N} \times \mathbb{N} \times 2^{\Sigma}$ of possible defects. Since Σ is finite and not empty, we assume that $\mathcal{D}$ is ordered such that for each $k \in \mathbb{N}$, the k^{th} element (x, y, H) of $\mathcal{D}$ is such that $x \leq k$. Then, for each $k \in \mathbb{N}$, we construct inductively a tuple $(U_k, \uparrow_k, h_k)$ where $U_k \subseteq \mathbb{N} \times \mathbb{N}$, $\uparrow_k \subseteq U_k \times U_k$ and $h_k : U_k \longrightarrow W$. The model $\mathcal{M}^{\mathrm{e}}$ is defined from all these tuples and the whole construction proceeds as follows:

Base case. Let $U_0 = \{0\} \times \mathbb{N}$, $\uparrow_0 = \varnothing$ and h_0 be such that for all $(0, y) \in U_0$, $h_0(0, y) = S^y(w)$.

Inductive case. Let $k > 0$ and suppose that $(U_k, \uparrow_k, h_k)$ has already been constructed. Let (x, y, H) be the k^{th} element of $\mathcal{D}$. If

(D1) $(x, y) \in U_k$,
(D2) $\Sigma(h_k(x, y)) \neq H$, and
(D3) there is $v \in W$ such that $h_k(x, y) \preccurlyeq v$ and $\Sigma(v) = H$,

then we construct $(U_{k+1}, \uparrow_{k+1}, h_{k+1})$ such that:

$$U_{k+1} = U_k \cup \{(c, d) \in \mathbb{N} \times \mathbb{N} \mid c = k+1 \text{ and } d \geq y\}$$
$$\uparrow_{k+1} = \uparrow_k \cup \{((a, b), (c, d)) \mid a = x, c = k+1, d \geq y \text{ and } b = d\}$$
$$h_{k+1} = h_k \cup \left\{((c, d), w) \;\middle|\; c = k+1, d \geq y \text{ and } w = S^{d-y}(v)\right\}$$

Otherwise $(U_{k+1}, \uparrow_{k+1}, h_{k+1}) = (U_k, \uparrow_k, h_k)$.

Final step. We construct $\mathcal{M}^{\mathrm{e}} = (W^{\mathrm{e}}, \preccurlyeq^{\mathrm{e}}, S^{\mathrm{e}}, V^{\mathrm{e}})$ such that $W^{\mathrm{e}} = \bigcup_{k \in \mathbb{N}} U_k$, $\preccurlyeq^{\mathrm{e}} = (\uparrow^{\mathrm{e}})^*$, where $\uparrow^{\mathrm{e}} = \bigcup_{k \in \mathbb{N}} \uparrow_k$, $S^{\mathrm{e}} = \{((a, b), (c, d)) \in W^{\mathrm{e}} \times W^{\mathrm{e}} \mid a = c \text{ and } d = b + 1\}$, and $V^{\mathrm{e}}(x, y) = V(h_x(x, y))$.

▶ **Lemma 5.** *For all $(x, y), (x', y') \in W^{\mathrm{e}}$, if $(x, y) \preccurlyeq^{\mathrm{e}} (x', y')$, then $x \leq x'$, $y = y'$ and $h_x(x, y) \preccurlyeq h_{x'}(x', y')$.*

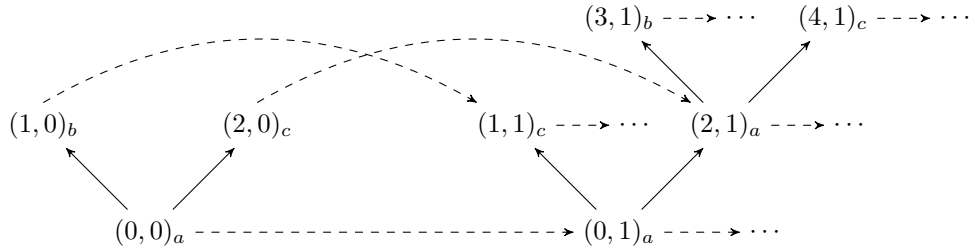


Figure 2 The strata W_0^{e}, W_1^{e} obtained from the model $\mathcal{M}$ defined in Figure 1. The subindexes indicate the value of $h = \bigcup_{k\in\mathbb{N}} h_k$.

Proof. Suppose that $(x,y) \preccurlyeq^{\mathrm{e}} (x',y')$. There is a sequence $(x_0,y_0)\dots(x_n,y_n)$ such that $(x_0,y_0) = (x,y)$, $(x_n,y_n) = (x',y')$ and for all $i < n$, $(x_i,y_i) \uparrow^{\mathrm{e}} (x_{i+1},y_{i+1})$. By construction, for all $i < n$, $(x_i,y_i) \uparrow_{x_{i+1}} (x_{i+1},y_{i+1})$ and $y_i = y_{i+1}$. Let $(x''_{i+1},y''_{i+1},H''_{i+1})$ be the $(x_{i+1}-1)^{\text{th}}$ element of $\mathcal{D}$ and v''_{i+1} the element of W choosen at the $(x_{i+1}-1)^{\text{th}}$ step. By construction, $x''_{i+1} = x_i$, $y''_{i+1} \leq y_i$ and by the ordering of $\mathcal{D}$, $x_i \leq x_{i+1}-1$. Moreover, $h_{x_i}(x_i,y''_{i+1}) \preccurlyeq v''_{i+1}$. Since $h_{x_i}(x_i,y_i) = S^{y_i-y''_{i+1}}\left(h_{x_i}\left(x_i,y''_{i+1}\right)\right)$ and $h_{x_{i+1}}(x_{i+1},y_{i+1}) = S^{y_{i+1}-y''_{i+1}}\left(v''_{i+1}\right)$, by the confluence condition for $\mathcal{M}$, $h_{x_i}(x_i,y_i) \preccurlyeq h_{x_{i+1}}(x_{i+1},y_{i+1})$. ◀

▶ **Lemma 6.** *$\mathcal{M}^{\mathrm{e}}$ is an expanding model.*

Proof. First we check that $\mathcal{M}^{\mathrm{e}}$ is a model. By Lemma 5, $\preccurlyeq^{\mathrm{e}}$ is antisymetric, hence a partial order. For the monotonicity condition, suppose that $(x,y) \preccurlyeq^{\mathrm{e}} (x',y')$. By Lemma 5, $h_x(x,y) \preccurlyeq h_{x'}(x',y')$ and by the monotonicity condition for $\mathcal{M}$, $V\left(h_x(x,y)\right) \subseteq V\left(h_{x'}(x',y')\right)$. For the confluence condition, it suffices to observe that by construction, if $(x,y) \uparrow^{\mathrm{e}} (x',y')$ then $(x,y+1) \uparrow^{\mathrm{e}} (x',y'+1)$. Therefore, $\mathcal{M}^{\mathrm{e}}$ is a model. To prove that $\mathcal{M}^{\mathrm{e}}$ is stratified, define $W^{\mathrm{e}}_n = \{(x,y) \in W^{\mathrm{e}} \mid y = n\}$ for all $n \in \mathbb{N}$. Conditions 3 of Def. 4 trivially holds and condition 1 comes directly from Lemma 5. To prove condition 2, it suffices to observe that by construction, for all $(x,y) \in W^{\mathrm{e}}$, either $x = 0$ or there is exactly one state $(x',y') \in W^{\mathrm{e}}$ such that $(x',y') \uparrow^{\mathrm{e}} (x,y)$. Therefore, by Lemma 5, for all $(x,y) \in W^{\mathrm{e}}$, there is a unique path from $(0,y)$ to (x,y). Finally, to prove that $\mathcal{M}^{\mathrm{e}}$ is expanding, suppose that $(c,b) \in W^{\mathrm{e}}$ and $(a,b+1) \uparrow^{\mathrm{e}} (c,b+1)$. Then the $(c-1)^{\text{th}}$ element of $\mathcal{D}$ is (a,y,H) for some y,H. Moreover, since $(c,b) \in W^{\mathrm{e}}$, $b \geq y$ and since $(a,y) \in W^{\mathrm{e}}$, $(a,b) \in W^{\mathrm{e}}$ and $(a,b) \uparrow^{\mathrm{e}} (c,b)$. Therefore it can easily be proved by induction on the length of the path from $S^{\mathrm{e}}(w)$ to $S^{\mathrm{e}}(v)$ that $S^{\mathrm{e}}(w) \preccurlyeq^{\mathrm{e}} S^{\mathrm{e}}(v)$ implies $w \preccurlyeq^{\mathrm{e}} v$. ◀

▶ **Lemma 7.** *For any state $(x,y) \in W^{\mathrm{e}}$ and any $\psi \in \Sigma$, $\mathcal{M}^{\mathrm{e}},(x,y) \vDash \psi$ if and only if $\mathcal{M},h_x(x,y) \vDash \psi$.*

Proof. The proof is by induction on the size $|\psi|$ of the formula. The cases for propositional variables, falsum, conjunctions and disjunctions are straightforward. For the temporal modalities, it suffices to observe that for all $(x,y) \in W^{\mathrm{e}}$ and all $n \in \mathbb{N}$, $(x,y+n) \in W^{\mathrm{e}}$ and $h_x(x,y+n) = S^n\left(h_x(x,y)\right)$. Finally, for implication, suppose first that $\mathcal{M}^{\mathrm{e}},(x,y) \nvDash \psi_1 \to \psi_2$. Then there is (x',y') such that $(x,y) \preccurlyeq^{\mathrm{e}} (x',y')$, $\mathcal{M}^{\mathrm{e}},(x',y') \vDash \psi_1$ and $\mathcal{M}^{\mathrm{e}},(x',y') \nvDash \psi_2$. By Lemma 5, $h_x(x,y) \preccurlyeq h_{x'}(x',y')$ and by induction hypothesis, $\mathcal{M},h_{x'}(x',y') \vDash \psi_1$ and $\mathcal{M},h_{x'}(x',y') \nvDash \psi_2$. Therefore, $\mathcal{M},h_x(x,y) \nvDash \psi_1 \to \psi_2$. For the other direction suppose that $\mathcal{M},h_x(x,y) \nvDash \psi_1 \to \psi_2$. There is $v' \in W$ such that $h_x(x,y) \preccurlyeq v'$, $\mathcal{M},v' \vDash \psi_1$ and $\mathcal{M},v' \nvDash \psi_2$. Let k be such that $(x,y,\Sigma(v'))$ is the k^{th} element of $\mathcal{D}$. Condition (D3) trivially holds and since $x \leq k$, condition (D1) holds too. Hence, there is $(x',y') \in W^{\mathrm{e}}$ such that

$\Sigma(h_{x'}(x', y')) = \Sigma(v')$ and either $(x', y') = (x, y)$ or $(x, y) \uparrow^{\mathrm{e}} (x', y')$. By induction hypothesis, $\mathcal{M}^{\mathrm{e}}, (x', y') \vDash \psi_1$ and $\mathcal{M}^{\mathrm{e}}, (x', y') \nvDash \psi_2$, hence $\mathcal{M}^{\mathrm{e}}, (x, y) \nvDash \psi_1 \to \psi_2$. ◀

In conclusion, we obtain the following:

▶ **Theorem 8.** *A formula φ is satisfiable (resp. falsifiable) on an intuitionistic dynamic model if and only if it is satisfiable (resp. falsifiable) on an expanding model.*

3.2 Persistent frames

Expanding models were introduced as a weakening of product models. They often lead to logics with a less complex validity problem. Thus it is natural to also consider a variant of $\mathsf{ITL}^{\mathsf{e}}$ interpreted over product models, or over the somewhat wider class of persistent models.

▶ **Definition 9.** Let $(W, \preccurlyeq)$ be a poset. If $S \colon W \to W$ is such that, whenever $v \succcurlyeq S(w)$, there is $u \succcurlyeq w$ such that $v = S(u)$, we say that S is *backward confluent*. If S is both forward and backward confluent, we say that it is *persistent*. A tuple $(W, \preccurlyeq, S)$ where S is persistent is a *persistent intuitionistic temporal frame*, and the set of valid formulas over the class of persistent intuitionistic temporal frames is denoted $\mathsf{ITL}^{\mathsf{p}}$, or *persistent domain* LTL.

The name 'persistent' comes from the fact that Theorem 8 can be modified to obtain a stratified model $\mathcal{M}'$ where $S' \colon W'_k \to W'_{k+1}$ is an isomorphism, i.e. whose domains are persistent with respect to S'. As we will see, the finite model property fails over the class of persistent models.

▶ **Lemma 10.** *The formula $\varphi = \neg\neg\Diamond\Box p \to \Diamond\neg\neg\Box p$ is not valid over the class of persistent models.*

Proof. Consider the model $\mathcal{M} = (W, \preccurlyeq, S, V)$, where $W = \mathbb{Z} \cup \{r\}$ with r a fresh world not in $\mathbb{Z}$, $w \preccurlyeq v$ if and only if $w = r$ or $w = v$, $S(r) = r$ and $S(n) = n + 1$ for $n \in \mathbb{Z}$, and $[\![p]\!] = [0, \infty)$. It is readily seen that $\mathcal{M}$ is a persistent model, that $\mathcal{M}, r \vDash \neg\neg\Diamond\Box p$ (since every maximal world above r satisfies $\Diamond\Box p$), yet $\mathcal{M}, r \nvDash \Diamond\neg\neg\Box p$, since there is no n such that $\mathcal{M}, S^n(r) \vDash \neg\neg\Box p$. It follows that $\mathcal{M}, r \nvDash \varphi$, and hence φ is not valid, as claimed. ◀

▶ **Lemma 11.** *The formula φ (from Lemma 10) is valid over the class of finite, persistent models.*

Proof. Let $\mathcal{M} = (W, \preccurlyeq, S, V)$ be a finite, persistent model, and assume that $\mathcal{M}, w \vDash \neg\neg\Diamond\Box p$. Let $v_1, \dots, v_n$ enumerate the maximal elements of $\{v \in W \mid w \preccurlyeq v\}$. For each $i \leq n$, let k_i be large enough so that $\mathcal{M}, S^{k_i}(v_i) \vDash \Box p$, and let $k = \max k_i$. We claim that $\mathcal{M}, S^k(w) \vDash \neg\neg\Box p$, which concludes the proof. Let $u \succcurlyeq S^k(w)$ be any leaf. Then, there is $v_i \succcurlyeq w$ such that $u = S^k(v_i)$ (since compositions of persistent functions are persistent). But, since $k \geq k_i$, we obtain $\mathcal{M}, u \vDash \Box p$, as desired. ◀

The following is then immediate from Lemmas 10 and 11:

▶ **Theorem 12.** $\mathsf{ITL}^{\mathsf{p}}$ *does not have the finite model property.*

Thus our decidability proof for $\mathsf{ITL}^{\mathsf{e}}$, which proceeds by first establishing a strong finite model property, does not carry over to $\mathsf{ITL}^{\mathsf{p}}$. Whether $\mathsf{ITL}^{\mathsf{p}}$ is decidable remains open.

4 Combinatorics of intuitionistic models

In this section we introduce some combinatorial tools we will need in order to prove that $\mathsf{ITL}^{\mathsf{e}}$ has the strong finite model property, and hence is decidable. We begin by discussing labeled structures, which allow for a graph-theoretic approach to intuitionistic models.

4.1 Labeled structures and quasimodels

▶ **Definition 13.** Given a set Λ whose elements we call 'labels' and a set W, a Λ-*labeling function* on W is any function $\lambda\colon W \to \Lambda$. A structure $\mathcal{S} = (W, R, \lambda)$ where W is a set, $R \subseteq W \times W$ and λ is a labeling function on W is a Λ-*labeled structure,* where 'structure' may be replaced with 'poset', 'directed graph', etc.

A useful measure of the complexity of a labeled poset or graph is given by its level:

▶ **Definition 14.** Given a labeled poset $\mathcal{A} = (W, \preccurlyeq, \lambda)$ and an element $w \in W$, an *increasing chain* from w of length n is a sequence $v_1 \dots v_n$ of elements of W such that $v_1 = w$ and $\forall i < n,\ v_i \prec v_{i+1}$, where $u \prec u'$ is shorthand for $u \preccurlyeq u'$ and $u' \not\preccurlyeq u$. The chain $v_1 \dots v_n$ is *proper* if it moreover satisfies $\forall i < n,\ \lambda(v_i) \neq \lambda(v_{i+1})$. The *depth* $\mathrm{dpt}(w) \in \mathbb{N} \cup \{\omega\}$ of w is defined such that $\mathrm{dpt}(w) = m$ if m is the maximal length of all the increasing chains from w and $\mathrm{dpt}(w) = \omega$ if there is no such maximum. Similarly, the *level* $\mathrm{lvl}(w) \in \mathbb{N} \cup \{\omega\}$ of w is defined such that $\mathrm{lvl}(w) = m$ if m is the maximal length of all the proper increasing chains from w and $\mathrm{lvl}(w) = \omega$ if there is no such maximum. The level $\mathrm{lvl}(\mathcal{A})$ of $\mathcal{A}$ is the maximal level of all its elements.

The notions of *depth* and *level* are extended to any acyclic directed graph $(W, \uparrow, \lambda)$ by taking the respective values on $(W, \uparrow^*, \lambda)$.

An important class of labeled posets comes from intuitionistic models.

▶ **Definition 15.** Given an intuitionistic Kripke model $\mathcal{M} = (W, \preccurlyeq, V)$ and a set $\Sigma \subseteq \mathcal{L}$ closed under subformulas, it can easily be checked that for all $w, v \in W$, if $w \preccurlyeq v$ then $\Sigma(w) \subseteq \Sigma(v)$. We denote the labeled poset $(W, \preccurlyeq, \Sigma(\cdot))$ by $\mathcal{M}^{\Sigma}$. Conversely, given a labeled poset $\mathcal{A} = (W, \preccurlyeq, \lambda)$ over 2^{Σ} such that if $w \preccurlyeq v$ then $\lambda(w) \subseteq \lambda(v)$, the valuation V_λ is defined such that $V_\lambda(w) = \{p \in \mathbb{P} \mid p \in \lambda(w)\}$ for all $w \in W$, and denote the resulting model by $\mathcal{A}^{\mathrm{mod}}$.

▶ **Definition 16.** Let Σ be a finite set of formulas closed under subformulas and $\mathcal{A} = (W, \preccurlyeq, \lambda)$ be a 2^{Σ}-labeled poset. We say that $\mathcal{A}$ is a Σ-*quasimodel* if λ is monotone in the sense that $w \preccurlyeq v$ implies that $\lambda(w) \subseteq \lambda(v)$, and whenever $\varphi \to \psi \in \Sigma$ and $w \in W$, we have that $\varphi \to \psi \in \lambda(w)$ if and only if, for all v such that $w \preccurlyeq v$, if $\varphi \in \lambda(v)$ then $\psi \in \lambda(v)$. If $(W, \preccurlyeq)$ is a tree, we say that $\mathcal{A}$ is *tree-like.*

4.2 Simulations, immersions and condensations

As is well-known, truth in intuitionistic models is preserved by bisimulation, and thus this is usually the appropriate notion of equivalence between different models. However, for our purposes, it is more convenient to consider a weaker notion, which we call *bimersion.*

▶ **Definition 17.** Given two labeled posets $\mathcal{A} = (W_{\mathcal{A}}, \preccurlyeq_{\mathcal{A}}, \lambda_{\mathcal{A}})$ and $\mathcal{B} = (W_{\mathcal{B}}, \preccurlyeq_{\mathcal{B}}, \lambda_{\mathcal{B}})$ and a relation $R \subseteq W_{\mathcal{A}} \times W_{\mathcal{B}}$, we write $\mathrm{dom}(R)$ for $\{w \in W_{\mathcal{A}} \mid \exists v \in W_{\mathcal{B}},\ (w, v) \in R\}$ and $\mathrm{rng}(R)$ for $\{v \in W_{\mathcal{B}} \mid \exists w \in W_{\mathcal{A}},\ (w, v) \in R\}$. A relation $\sigma \subseteq W_{\mathcal{A}} \times W_{\mathcal{B}}$ is a *simulation* from $\mathcal{A}$ to $\mathcal{B}$

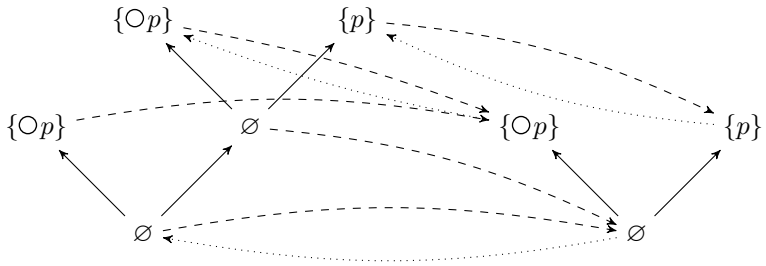


Figure 3 A condensation from the labeled frame on the left to the labeled frame on the right. Dashed arrows indicate ρ, dotted arrows ι.

if $\mathrm{dom}(\sigma) = W_{\mathcal{A}}$ and whenever $w\ \sigma\ v$, it follows that $\lambda_{\mathcal{A}}(w) = \lambda_{\mathcal{B}}(v)$, and if $w \preccurlyeq_{\mathcal{A}} w'$ then there is v' so that $v \preccurlyeq_{\mathcal{B}} v'$ and $w'\ \sigma\ v'$.

A simulation is called an *immersion* if it is a function. If an immersion $\sigma\colon W_{\mathcal{A}} \to W_{\mathcal{B}}$ exists, we write $\mathcal{A} \trianglelefteq \mathcal{B}$. If, moreover, there is an immersion $\tau\colon W_{\mathcal{B}} \to W_{\mathcal{A}}$, we say that they are *bimersive*, write $\mathcal{A} \triangleq \mathcal{B}$, and call the pair (σ, τ) a *bimersion*. A *condensation from* $\mathcal{A}$ *to* $\mathcal{B}$ is a bimersion (ρ, ι) so that $\rho\colon W_{\mathcal{A}} \to W_{\mathcal{B}}$, $\iota\colon W_{\mathcal{B}} \to W_{\mathcal{A}}$, ρ is surjective, and $\rho\iota$ is the identity on $W_{\mathcal{B}}$. If such a condensation exists we write $\mathcal{B} \ll \mathcal{A}$. Observe that $\mathcal{B} \ll \mathcal{A}$ implies that $\mathcal{B} \triangleq \mathcal{A}$.

If $\mathcal{M}, \mathcal{N}$ are models and Σ a set of formulas closed under subformulas, we write $\mathcal{M} \trianglelefteq_\Sigma \mathcal{N}$ if $\mathcal{M}^\Sigma \trianglelefteq \mathcal{N}^\Sigma$, and define $\triangleq_\Sigma, \ll_\Sigma$ similarly. We may also write e.g. $\mathcal{A} \ll \mathcal{M}$ if $\mathcal{A}$ is 2^Σ-labeled and $\mathcal{A} \ll \mathcal{M}^\Sigma$.

In this text, simulations will *always* be between posets, and if instead $\mathcal{A}$ or $\mathcal{B}$ is an acyclic directed graph, a simulation between $\mathcal{A}$ and $\mathcal{B}$ will be one between their respective transitive closures. It will typically be convenient to work with immersions rather than simulations: however, as the next lemma shows, not much generality is lost by this restriction.

▶ **Lemma 18.** *Let $\mathcal{A} = (W_{\mathcal{A}}, \preccurlyeq_{\mathcal{A}}, \lambda_{\mathcal{A}})$ and $\mathcal{B} = (W_{\mathcal{B}}, \preccurlyeq_{\mathcal{B}}, \lambda_{\mathcal{B}})$ be labeled posets. If a simulation $\sigma \subseteq W_{\mathcal{A}} \times W_{\mathcal{B}}$ exists, $W_{\mathcal{A}}$ is a finite tree, and $w\ \sigma\ w'$, then there is an immersion $\sigma' \subseteq W_{\mathcal{A}} \times W_{\mathcal{B}}$ such that $w' = \sigma'(w)$.*

Proof. By a straightforward induction on the depth of w we show that there is a partial immersion σ_w with $w \in \mathrm{dom}(\sigma_w)$, whose domain is the subtree generated by w, and such that $w\ \sigma\ w'$. Let D be the set of childrenset of daughters of w, and for each $v \in D$, choose v' so that $v\ \sigma\ v'$ and $w' \preccurlyeq_{\mathcal{B}} v'$. By the induction hypothesis, there is a partial immersion σ'_v with $v \in \mathrm{dom}(\sigma'_v)$. Then, one readily checks that $\{(w, w')\} \cup \bigcup_{v \in D} \sigma'_v$ is also an immersion, as needed. ◀

Condensations are useful for producing (small) quasimodels out of models.

▶ **Proposition 19.** *Given an intuitionistic model $\mathcal{M} = (W_{\mathcal{M}}, \preccurlyeq_{\mathcal{M}}, V_{\mathcal{M}})$, a set Σ of intuitionistic formulas that is closed for subformulas, and a 2^Σ-labeled poset $\mathcal{A} = (W_{\mathcal{A}}, \preccurlyeq_{\mathcal{A}}, \lambda_{\mathcal{A}})$ over Σ, if $\mathcal{A} \ll \mathcal{M}$, then $\mathcal{A}$ is a quasimodel.*

Proof. See Appendix. ◀

4.3 Normalized labeled trees

In order to count the number of different labeled trees up to bimersion, we construct, for any set Λ of labels and any $k \geq 1$, the labeled directed acyclic graph $\mathcal{G}_k^\Lambda = \left(W_k^\Lambda, \uparrow_k^\Lambda, \lambda_k^\Lambda\right)$ by induction on k as follows.

Base case. For $k = 1$, let $\mathcal{G}_1^\Lambda = \left(W_1^\Lambda, \uparrow_1^\Lambda, \lambda_1^\Lambda\right)$ with $W_1^\Lambda = \Lambda W_1^\Lambda = L$, $\uparrow_1^\Lambda = \varnothing$, and $\lambda_1^\Lambda(w) = w$ for all $w \in W_1^\Lambda$.

Inductive case. Suppose that $\mathcal{G}_k^\Lambda = \left(W_k^\Lambda, \uparrow_k^\Lambda, \lambda_k^\Lambda\right)$ has already been defined. The graph $\mathcal{G}_{k+1}^\Lambda = \left(W_{k+1}^\Lambda, \uparrow_{k+1}^\Lambda, \lambda_{k+1}^\Lambda\right)$ is constructed such that:

$$W_{k+1}^\Lambda = W_k^\Lambda \cup P$$
$$\uparrow_{k+1}^\Lambda = \uparrow_k^\Lambda \cup \left\{(x,y) \in W_{k+1}^\Lambda \times W_{k+1}^\Lambda \;\middle|\; \exists (\ell, C) \in P,\ x = (\ell, C) \text{ and } y \in C\right\}$$
$$\lambda_{k+1}^\Lambda(w) = \begin{cases} \lambda_k^\Lambda(w) & \text{if } w \in W_k^\Lambda \\ \ell & \text{if } w = (\ell, C) \in P \end{cases}$$

where $P = \left\{(\ell, C) \in \Lambda \times \mathcal{P}\left(W_k^\Lambda\right) \;\middle|\; \forall y \in C,\ \lambda_k^\Lambda(y) \neq \ell\right\}$.

Note that $\mathcal{G}_k^\Lambda = \left(W_k^\Lambda, \uparrow_k^\Lambda, \lambda_k^\Lambda\right)$ is typically not a tree, but we may unravel it to obtain one.

▶ **Definition 20.** Given a labeled directed acyclic graph $\mathcal{G} = (W, \uparrow, \lambda)$ and a node $w \in W$, the *unraveling* of $\mathcal{G}$ from w is the labeled tree $\mathcal{T}_w = (W_w, \uparrow_w, \lambda_w)$ such that W_w is the set of all the paths from w in $\mathcal{G}$, $\xi \uparrow_w \zeta$ if and only if there is $v \in W$ such that $\zeta = \xi v$, and $\lambda_w(v_0 \dots v_n) = \lambda(v_n)$.

▶ **Proposition 21.** *For any rooted labeled tree $\mathcal{T} = (W, \uparrow, \lambda)$ over a set Λ of labels, if the level of $\mathcal{T}$ is finite then there is a condensation from $\mathcal{T}$ to an unraveling of $\mathcal{G}_{\mathrm{lvl}(\mathcal{T})}^\Lambda = \left(W_{\mathrm{lvl}(\mathcal{T})}^\Lambda, \uparrow_{\mathrm{lvl}(\mathcal{T})}^\Lambda, \lambda_{\mathrm{lvl}(\mathcal{T})}^\Lambda\right)$.*

Proof. Let $\mathcal{T} = (W, \uparrow, \lambda)$ be a labeled treedirected acyclic graph with root r. We write $\prec$ for the transitive closure of $\uparrow$ and $\preccurlyeq$ for the reflexive closure of $\prec$. The proof is by induction on the level $n = \mathrm{lvl}(\mathcal{T})$ of $\mathcal{T}$. For $n = 1$, observe that this means that $\lambda(w) = \lambda(r)$ for all $w \in W$. Let $\rho = W \times \{\lambda(r)\}$ and $\iota = \{(\lambda(r), r)\}$. It can easily be checked that (ρ, ι) is a condensation.[2] For $n > 1$, suppose the property holds for all rooted labeled trees $\mathcal{T}'$ such that $\mathrm{lvl}(\mathcal{T})' < n$. Define the following sets:

$$N = \{w \in W \mid \lambda(w) \neq \lambda(r) \text{ and for all } v \prec w,\ \lambda(v) = \lambda(r)\}$$
$$M = \{w \in W \mid \text{for all } v \preccurlyeq w,\ \lambda(v) = \lambda(r)\}$$

Clearly, for all $w \in N$, $\mathrm{lvl}(w) < n$. Therefore, by induction, there is a condensation (ρ_w, ι_w) from the subgraph of $\mathcal{T}$ generated by w to the unraveling of $\mathcal{G}_{n-1}^\Lambda$ from some $y_w \in W_{n-1}^\Lambda$. Let us define $r' = (\lambda(r), \{y_w \mid w \in N\})$ and consider the unraveling $\mathcal{G}$ of $\mathcal{G}_n^\Lambda$ from r'. It can easily be checked that $\rho = (M \times \{r'\}) \cup \bigcup_{w \in N} \rho_w \rho = (S \times \{r'\}) \cup \bigcup_{w \in W} \rho_w$ is an immersion from $\mathcal{T}$ to $\mathcal{G}$, $\iota' = \{(r', r)\} \cup \bigcup_{w \in N} \iota_w \iota' = \{(r', r)\} \cup \bigcup_{w \in W} \iota_w$ is a simulation from $\mathcal{G}$ to $\mathcal{T}$ and $\iota' \subseteq \rho^{-1}$. Using Lemma 18, we can then choose an immersion $\iota \subseteq \iota'$, so that (ρ, ι) is a condensation from $\mathcal{T}$ to $\mathcal{G}$. ◀

Finally, let us define recursively E_k^n and Q_k^n for all $n, k \in \mathbb{N}$ by:

$$E_k^n = \begin{cases} 0 & \text{if } k = 0 \\ E_{k-1}^n + n2^{E_{k-1}^n} & \text{otherwise} \end{cases} \qquad Q_k^n = \begin{cases} 0 & \text{if } k = 0 \\ 1 + E_{k-1}^n Q_{k-1}^n & \text{otherwise} \end{cases}$$

[2] Recall that as per our convention, this means that (ρ, ι) is a condensation between the respective transitive closures.

The following lemma can be proven by a straightforward induction, left to the reader.

▶ **Lemma 22.** *For any finite set Λ with cardinality n and all $k \in \mathbb{N}$,*
1. *the cardinality of $\mathcal{G}_k^\Lambda$ is bounded by E_k^n, and*
2. *the cardinality of any unraveling of $\mathcal{G}_k^\Lambda$ is bounded by Q_k^n.*

From these and Proposition 21, we obtain the following:

▶ **Theorem 23.**
1. *Given a set of labels Λ and a Λ-labeled tree $\mathcal{T}$ of level $k < \omega$, there is a Λ-labeled tree $\mathcal{T}'$ bounded by $Q_k^{|\Lambda|}$ such that $\mathcal{T}' \triangleq \mathcal{T}$. We call $\mathcal{T}'$ the* normalized Λ-labeled tree for $\mathcal{T}$.
2. *Given a sequence of Λ-labeled trees $\mathcal{T}_1, \dots, \mathcal{T}_n$ of level $k < \omega$ with $n > E_k^{|\Lambda|}$, there are indexes $i < j \leq n$ such that $\mathcal{T}_i \triangleq \mathcal{T}_j$.*

The second item may be viewed as a finitary variant of Kruskal's theorem for labeled trees [19]. When applied to quasimodels, we obtain the following:

▶ **Proposition 24.** *Let Σ be a set of formulas closed under subformulas with $|\Sigma| = s < \omega$.*
1. *Given a tree-like Σ-quasimodel $\mathcal{T}$ and $s = |\Sigma|$, there is a tree-like Σ-quasimodel $\mathcal{T}' \triangleq_\Sigma \mathcal{T}$ bounded by $Q_{s+1}^{2^s}$. We call $\mathcal{T}'$ the* normalized Σ-quasimodel for $\mathcal{T}$.
2. *Given a sequence of tree-like Σ-quasimodels $\mathcal{T}_1, \dots, \mathcal{T}_n$ with $n > E_{s+1}^{2^s}$, there are indexes $i < j \leq n$ such that $\mathcal{T}_i \triangleq \mathcal{T}_j$.*

Proof. Immediate from Proposition 19 and Theorem 23Lemma 22 using the fact that any Σ-quasimodel has level at most $s + 1$. ◀

Finally, we obtain an analogous result for pointed structures.

▶ **Definition 25.** A *pointed labeled poset* is a structure $(W, \preccurlyeq, \lambda, w)$ consisting of a labeled tree with a designated world $w \in W$. Given a labeled poset $\mathcal{A} = (W_\mathcal{A}, \preccurlyeq_\mathcal{A}, \lambda_\mathcal{A})$ and $w \in W_\mathcal{A}$, we denote by $\mathcal{A}^w$ the pointed labeled poset given by $\mathcal{A}^w = (W_\mathcal{A}, \preccurlyeq_\mathcal{A}, \lambda_\mathcal{A}, w)$. A *pointed simulation* between pointed labeled posets $\mathcal{A} = (W_\mathcal{A}, \preccurlyeq_\mathcal{A}, \lambda_\mathcal{A}, w_\mathcal{A})$ and $\mathcal{B} = (W_\mathcal{B}, \preccurlyeq_\mathcal{B}, \lambda_\mathcal{B}, w_\mathcal{B})$ is a simulation $\sigma \subseteq W_\mathcal{A} \times W_\mathcal{B} \sigma \subset W_\mathcal{A} \times W_\mathcal{B}$ such that if $w \; \sigma \; v$, then $w = w_\mathcal{A}$ if and only if $v = w_\mathcal{B}$. The notions of *pointed immersion, pointed condensation,* etc. are defined analogously to Definition 17.

▶ **Lemma 26.** *If Λ has n elements, any pointed Λ-labeled poset of level at most k condenses to a labeled pointed tree bounded by Q_{k+2}^{2n}, and there are at most E_{k+2}^{2n} bimersion classes.*

Proof. We may view a pointed labeled poset $\mathcal{A} = (W, \preccurlyeq, \lambda, w)$ as a (non-pointed) labeled poset as follows. Let $\Lambda' = \Lambda \times \{0, 1\}$. Then, set $\lambda'(v) = (\lambda(v), 0)$ if $v \neq w$, $\lambda'(w) = (\lambda(w), 1)$. Note that $\mathcal{A}$ may now have level $k+2$, since we may have that $u \preccurlyeq w \preccurlyeq v$, $\lambda(u) = \lambda(w) = \lambda(v)$, yet $\lambda'(u) \neq \lambda'(w)$ and $\lambda'(w) \neq \lambda'(v)$. By Proposition 21, $\mathcal{A}$ condenses to a generated tree $\mathcal{T}$ of $\mathcal{G}_{k+2}^{\Lambda'}$ by some condensation (ρ, ι). Let $w' = \rho(w)$, and consider $\mathcal{T}$ as a pointed structure with distinguished point w'. Given that ρ is a surjective, label-preserving function, w, w' are the only points whose label has second component 1, and therefore (ρ, ι) must be a pointed condensation, as claimed. ◀

▶ **Proposition 27.** *Let Σ be a set of formulas closed under subformulas with $|\Sigma| = s < \omega$.*
1. *Given a tree-like pointed Σ-quasimodel $\mathcal{T}$ and a formula φ, there is a tree-like pointed Σ-quasimodel $\mathcal{T}' \triangleq \mathcal{T}$ bounded by $Q_{s+3}^{2^{s+1}}$. We call $\mathcal{T}'$ the* normalized pointed Σ-quasimodel for $\mathcal{T}$.

2. *Given a sequence of tree-like pointed Σ-quasimodels $\mathcal{T}_1, \ldots, \mathcal{T}_n$ with $n > E_{s+3}^{2^{s+1}}$, there are indexes $i < j \leq n$ such that $\mathcal{T}_i \triangleq \mathcal{T}_j$.*

With these tools at hand, we are ready to prove that $\mathsf{ITL}^{\mathsf{e}}$ has the strong finite model property, and hence is decidable.

5 Decidability

The following transformations are defined for any stratified model $\mathcal{M} = (W, \preccurlyeq, S, V)$ and any finite, non-empty set of formulas Σ closed under subformulas. In each case, given a stratified model $\mathcal{M} = (W, \preccurlyeq, S, V)$, we will produce another stratified model $\mathcal{M}' = (W', \preccurlyeq', S', V')$ and a map $\pi \colon W' \to W$ such that $\Sigma_{\mathcal{M}}(\pi(w)) = \Sigma_{\mathcal{M}'}(w)$ for all $w \in W' \Sigma_{\mathcal{M}}(w) = \Sigma_{\mathcal{M}'}(\pi(w))$ for all $w \in W'$.

Replace $\mathcal{M}_k$ with a copy of the normalized Σ-quasimodel of $\mathcal{M}_k$. Let $\mathcal{T} = (W_{\mathcal{T}}, \uparrow_{\mathcal{T}}, \lambda_{\mathcal{T}})$ be a copy of the normalized labeled tree of $\mathcal{M}_k^{\Sigma}$ such that $W_{\mathcal{T}} \cap W = \varnothing$, and (ρ, ι) the condensation from $\mathcal{M}_k^{\Sigma}$ to $\mathcal{T}$. The result of the transformation is the tuple $(W', \preccurlyeq', S', V')$ such that $W' = W \cup W_{\mathcal{T}} \setminus W_k$, $\preccurlyeq' = \preccurlyeq\restriction_{W \setminus W_k} \cup (\uparrow_{\mathcal{T}})^*$,

$$S'(w) = \begin{cases} \rho(S(w)) & \text{if } w \in W_{k-1} \\ S(\iota(w)) & \text{if } w \in W_{\mathcal{T}} \\ S(w) & \text{otherwise} \end{cases} \qquad V'(w) = \begin{cases} \{p \mid p \in \lambda_T(w)\} & \text{if } w \in W_{\mathcal{T}} \\ V(w) & \text{otherwise} \end{cases}$$

The map π is the identity on $W_i' = W_i$ for $i \neq k$, and $\pi(w) = \iota(w)$ for $w \in W_{\mathcal{T}}$.

Replace $\mathcal{M}_k$ with a copy of the normalized, pointed Σ-quasimodel of $\mathcal{M}_k$ preserving w, where $w \in W_k$. The transformation is similar to the previous one except that $\mathcal{M}_k$ is regarded as a pointed structure with distinguished point w.

Replace $\mathcal{M}_\ell$ with $\mathcal{M}_k$, where $k < \ell$ and there is an immersion $\sigma \colon W_k \to W_\ell$ (seen as 2^{Σ}-labeled trees). The result of the transformation is the tuple $(W', \preccurlyeq', S', V')$ such that $W' = W \setminus \bigcup_{k < m \leq \ell} W_m$, $\preccurlyeq' = \preccurlyeq\restriction_{W'}$,

$$S'(w) = \begin{cases} S(\sigma(w)) & \text{if } S(w) \in W_k \\ S(w) & \text{otherwise} \end{cases}$$

and $V' = V\restriction_{W'}$.

The map π is the identity on $W_i' = W_i$ for $i < k$, on $W_i' = W_{i+\ell-k}$ for $i > k$, and $\pi(w) = \sigma(w)$ for all $w \in W_k'$.

Replace $\mathcal{M}_\ell$ with $\mathcal{M}_k$ connecting w_k to w_ℓ, where $k < \ell$, $w_k \in W_k$, $w_\ell \in W_\ell$ and there is an immersion $\sigma \colon W_k \to W_\ell$ such that $\sigma(w_k) = w_\ell$. The transformation is defined as the previous one.

▶ **Lemma 28.** *The result of any previous transformation is a stratified model such that $\Sigma_{\mathcal{M}}(\pi(w)) = \Sigma_{\mathcal{M}'}(w) \Sigma_{\mathcal{M}}(w) = \Sigma_{\mathcal{M}'}(\pi(w))$ for any $w \in W'$.*

Proof. The proof that $\mathcal{M}' = (W', \preccurlyeq', S', V')$ is a stratified modela model is straightforward and left to the reader. We prove by structural induction on φ that for all transformations,

all $w \in W'$ and all $\varphi \in \Sigma$, $\mathcal{M}', w \vDash \varphi$ iff $\mathcal{M}, \pi(w) \vDash \varphi$. We only detail the case for the next modality when $\mathcal{M}_k$ is replaced with a copy of the normalized Σ-quasimodel $\mathcal{T}$ of $\mathcal{M}_k$ and $w \in W_{k-1}w \in W'_{k-1}$. The cases for the other temporal modalities are similar (see also the proof of Lemma 29). The cases for the implication are similar as in the proof of Proposition 19. The remaining cases are straighforward. Suppose that $w \in W_{k-1}w \in W'_{k-1}$ and $\mathcal{M}, \pi(w) \vDash \bigcirc\psi$. Then $\psi \in \Sigma(S(w))$. Since $S'(w) = \rho(S(w))$, $\pi(S'(w)) = \iota(S'(w))$ and (ρ, ι) is a condensation, $\Sigma(S(w)) = \lambda_{\mathcal{T}}(S'(w)) = \Sigma(\pi(S'(w)))$. Since $\mathcal{M}, \pi(S'(w)) \vDash \psi$, by induction hypothesis, $\mathcal{M}', S'(w) \vDash \psi$. Hence $\mathcal{M}', w \vDash \bigcirc\psi$.and $\mathcal{M}, \pi(S'(w)) \vDash \psi$. By induction hypothesis, $\mathcal{M}', S'(w) \vDash \psi$, hence $\mathcal{M}', w \vDash \bigcirc\psi$. The other direction is similar. ◀

Now, let us consider a stratified model $\mathcal{M} = (W, \preccurlyeq, S, V)$ with w_0 the root of W_0. The finite model $\mathcal{M}^{\text{fin}} = \left(W^{\text{fin}}, \preccurlyeq^{\text{fin}}, S^{\text{fin}}, V^{\text{fin}}\right)$ with a state w_0^{fin} such that $\Sigma(w_0^{\text{fin}}) = \Sigma(w_0)$ is constructed by the following procedure. This procedure is in three phases plus a final step. At each step, the model $\mathcal{M}$ is modified in place. Moreover, three index variables are maintained by the procedure:

- The variable i, initialized to 0, indicates the current labeled tree W_i which is considered.
- The variable j, initially undefined, indicates the index of the first labeled tree occuring infinitely often up to bimersion.
- The variable ℓ, initially undefined, holds the index of the last labeled tree that must not be modified.

As an invariant, $\mathcal{M}$ is stratified until the final step and for all $k < i$, $\mathcal{M}_k$ is a copy of a normalized labeled tree.

First phase

- If there is $k < i$ such that $\mathcal{M}_k \trianglelefteq \mathcal{M}_i$, replace $\mathcal{M}_i$ with $\mathcal{M}_k$, set i to $k+1$ and redo the same phase.
- If not, and for all $x > i$ there is $y > x$ such that $\mathcal{M}_y \trianglelefteq \mathcal{M}_i$, then replace $\mathcal{M}_i$ with a copy of its normalized Σ-quasimodel, increase i by one, set j and ℓ to i and start the next phase.
- Otherwise, replace $\mathcal{M}_i$ with its normalized Σ-quasimodel, increase i by one and redo the same phase.

Second phase. In this phase, we need to care about eventualities. To this end, a current eventuality (w, ψ), initially undefined, is maintained across the executions of the phase. Let w_x denote the element of the fulfillment of (w, ψ) belonging to W_x (if it exists), and $\mathcal{M}_x^+$ be the pointed structure $\mathcal{M}_x^{w_x}$. The phase proceeds through the following steps:

- If (w, ψ) is defined and the last element of the fulfillment of (w, ψ) belongs to some W_k with $k \leq i$ then undefine (w, ψ), set ℓ to i and repeat the same phase.
- If (w, ψ) is undefined then choose an eventuality (w, ψ) such that $w \in W_j$ and the last element of its fulfillment belongs to some W_k with $k > i$ and we and repeat the same phaseMissing text. If there is no such eventuality then start the next phase.
- If (w, ψ) is defined and there is k such that $\ell < k < i$ and $\mathcal{M}_k^+ \trianglelefteq \mathcal{M}_i^+$, then replace $\mathcal{M}_i$ with $\mathcal{M}_k$ connecting w_k to w_i, set i to $k+1$ and redo the same phase.
- Otherwise, replace $\mathcal{M}_i$ with a copy of the normalized labeled tree of $\mathcal{M}_k$ preserving w_i, increase i and redo the same phase.

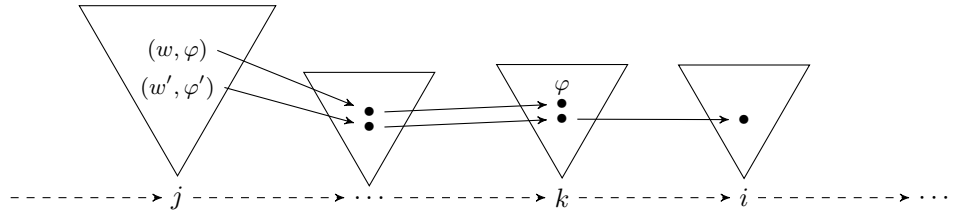


Figure 4 The stratum $\mathcal{M}_j$ and two of its eventualities. The fulfillment of (w, φ) is displayed, as well as the initial portion of the fulfillment of (w', φ').

Third phase

- If $\mathcal{M}_i \trianglelefteq \mathcal{M}_j$, then start the final step.
- If there is k such that $\ell < k < i$ and $\mathcal{M}_k \trianglelefteq \mathcal{M}_i$, then replace $\mathcal{M}_i$ with $\mathcal{M}_k$, set i to $k+1$ and redo the same phase.
- Otherwise, replace $\mathcal{M}_i$ with a copy of its normalized Σ-quasimodel, increase i by one and redo the same phase.

Final step. There is an immersion $\sigma\colon W_i \to W_j$. Construct the final tuple $(W^{\text{fin}}, \preccurlyeq^{\text{fin}}, S^{\text{fin}}, V^{\text{fin}})$ such that $W^{\text{fin}} = \bigcup_{0 \le m < i} W_m$, $\preccurlyeq^{\text{fin}} = \preccurlyeq\restriction_{W^{\text{fin}}}$,

$$S^{\text{fin}}(w) = \begin{cases} \sigma\left(S\left(w\right)\right) & \text{if } w \in W_{i-1} \\ S(w) & \text{otherwise} \end{cases}$$

$V^{\text{fin}} = V\restriction_{W^{\text{fin}}}$, and w_0^{fin} is the root of W_0 (note that $w_0^{\text{fin}} \in W^{\text{fin}}$).

▶ **Lemma 29.** *The final tuple is a model and* $\Sigma(w_0^{\text{fin}}) = \Sigma(w_0)$.

Proof. The proof that $\mathcal{M}^{\text{fin}} = \left(W^{\text{fin}}, \preccurlyeq^{\text{fin}}, S^{\text{fin}}, V^{\text{fin}}\right)$ is a model is straightforward and left to the reader. We prove by structural induction on φ that for all $w \in W^{\text{fin}}$ and all $\varphi \in \Sigma$, $\mathcal{M}^{\text{fin}}, w \vDash \varphi$ iff $\mathcal{M}, w \vDash \varphi$. The cases for propositional variables and the boolean connectives are straightforward. The case for the next temporal modality is similar as in the proof of Lemma 28. For the eventually and henceforth temporal modalities, suppose first that (w, φ) is an eventuality in $\mathcal{M}$ and $w \in W^{\text{fin}}$. Let $w_0 \dots w_n$ be the fulfillment of (w, φ) in $\mathcal{M}$. If $w_n \in W^{\text{fin}}$ then by induction hypothesis, (w, φ) is an eventuality in $\mathcal{M}^{\text{fin}}$. Otherwise, there is $k \le n$ such that $w_k \in W_i$. Therefore, (w_k, φ) is an eventuality in $\mathcal{M}$ and so is $(\sigma(w_k), \varphi)$. Since by construction, after the second phase, the length of the fulfillment of any eventuality (v, φ) such that $v \in W_j$ is bounded by $1 + i - j$, (w, φ) is an eventuality in $\mathcal{M}^{\text{fin}}$. Conversely, suppose now that (w, φ) is an eventuality in $\mathcal{M}^{\text{fin}}$ and let $w_0 \dots w_n$ be its fulfillment. For each $k \le n$ let m_k be such that $w_k \in W_{m_k}$. The proof is by a subinduction on the number r of $k \in \{1\,..\,n\} k \in 1\,..\,n$ such that $m_k = j$. If $r = 0$ then by induction hypothesis, (w, φ) is an eventuality in $\mathcal{M}$. If $r > 0$, let $k > 0$ be the least index such that $m_k = j$. If $k = n$ then suppose that $\varphi = \Diamond\psi$, the other case being symmetric. We have the other case beeing symmetric When have$\mathcal{M}^{\text{fin}}, w_k \vDash \psi$ and by induction $\mathcal{M}, w_k \vDash \psi$. Since $k > 0$, $w_k = S^{\text{fin}}(w_{k-1}) = \sigma(S(w_{k-1}))$ and since σ is an immersion, $\mathcal{M}, S(w_{k-1}) \vDash \psi$. Therefore (w, φ) is an eventuality in $\mathcal{M}$. Finally, if $r > 0$ and $k < n$ then (w_k, φ) is an eventuality in $\mathcal{M}^{\text{fin}}$ and by the subinduction hypothesis (w_k, φ) is an eventuality in $\mathcal{M}$. Since $k > 0$, $w_k = S^{\text{fin}}(w_{k-1}) = \sigma(S(w_{k-1}))$. MoreoverMorevoer, since σ is an immersion, $(S(w_{k-1}), \varphi)$ is an eventuality in $\mathcal{M}$. Hence (w, φ) is an eventuality in $\mathcal{M}$. ◀

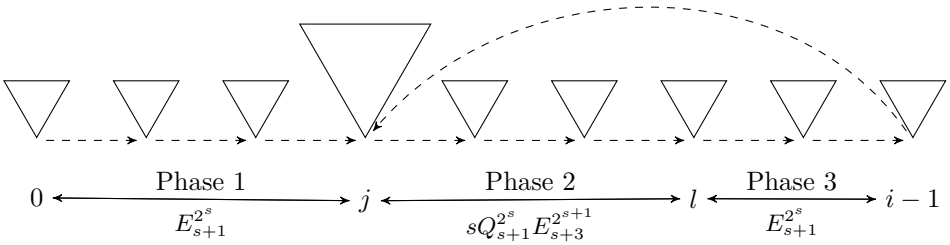


Figure 5 An illustration of the three phases of $\mathcal{M}^{\text{fin}}$. Below each phase we indicate the number of strata, used for the computations in the proof of Lemma 30.

▶ **Lemma 30.** *The cardinality of* W^{fin} *is bounded by*

$$B(s) \stackrel{def}{=} Q^{2^{s+1}}_{s+3}\left(2E^{2^s}_{s+1} + sQ^{2^s}_{s+1}E^{2^{s+1}}_{s+3}\right)$$

where $s = |\Sigma|$.

Proof. See Appendix. ◀

We have proved the following strong finite model property.

▶ **Theorem 31.** *There exists a computable function* B *such that for any formula* $\varphi \in \mathcal{L}$*, if* φ *is satisfiable (resp. unsatisfiable) then* φ *is satisfiable (resp. falsifiable) in a model* $\mathcal{M} = (W, \preccurlyeq, S, V)$ *such that* $|W| \leq B(|\varphi|)$.

Proof. In view of Theorem 8, a formula φ is satisfiable (resp. falsifiable) in a model $\mathcal{M}$ if and only if it is satisfied (resp. falsified) at the root of a stratified model $\mathcal{M}^{\text{e}}$. Then, by Lemma 29, φ is satisfied (resp. falsified) in $\mathcal{M}^{\text{e}}$ if and only if it is satisfied (res. falsified) on $(\mathcal{M}^{\text{e}})^{\text{fin}}$, which is effectively bounded by $B(|\varphi|)$ by Lemma 30. ◀

As a corollary, we get the decidability of ITL^{e}.

▶ **Corollary 32.** *The satisfiability and validity problems for* ITL^{e} *are decidable.*

6 Conclusion

We have introduced ITL^{e}, an intuitionistic analogue of LTL based on expanding domain models from modal logic. In the literature, intuitionistic modal logic is typically interpreted over persistent models, but as we have shown this interpretation has the technical disadvantage of not enjoying the finite model property. Of course, this fact alone does not imply that ITL^{p} is undecidable, and whether the latter is true remains an open problem. Meanwhile, our semantics are natural in the sense that we impose the minimal conditions on S so that all truth values are upwards closed under $\preccurlyeq$, and a wider class of models is convenient as they can more easily be tailored for specific applications.

This is an exploratory work, being the first to consider the logic ITL^{e}. As can be gathered from the tools we have developed, understanding this logic poses many technical challenges, and many interesting questions remain open. Perhaps the most pressing is the complexity of validity and satisfiability: the decision procedure we have given is non-elementary, but there seems to be little reason to assume that this is optimal. It may be possible to further 'trim' the model $\mathcal{M}^{\text{fin}}$ to obtain one that is elementarily bounded. However, we should not expect

polynomially bounded models, as $\mathsf{ITL}^{\mathsf{e}}$ is conservative over intuitionistic propositional logic, which is already PSPACE-complete. Finally, we leave open the problem of finding a sound and complete axiomatization for $\mathsf{ITL}^{\mathsf{e}}$.

Acknowledgements. We would like to thank the anonymous referees for their useful suggestions which helped improve the presentation.

References

1 J.-M. Alliot, M. Diéguez, and L. Fariñas del Cerro. Metabolic pathways as temporal logic programs. In *Logics in Artificial Intelligence – 15th European Conference, JELIA 2016, Larnaca, Cyprus, November 9–11, 2016, Proceedings*, pages 3–17, 2016.

2 P. Balbiani and M. Diéguez. Temporal here and there. In M. Loizos and A. Kakas, editors, *Logics in Artificial Intelligence*, pages 81–96. Springer, 2016.

3 G. Boenn, M. Brain, M. De vos, and J. Ffitch. Automatic music composition using answer set programming. *Theory Pract. Log. Program.*, 11(2-3):397–427, 2011.

4 P. Cabalar and M. Diéguez. Strong Equivalence of Non-Monotonic Temporal Theories. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 14th International Conference, KR'14, Vienna, Austria*, July 20–24, 2014.

5 P. Cabalar and G. Pérez. Temporal Equilibrium Logic: A First Approach. In *EUROCAST'07*, page 241–248, Las Palmas de Gran Canaria, Spain, 2007.

6 D. Van Dalen. Intuitionistic logic. In *Handbook of Philosophical Logic*, volume 166, pages 225–339. Springer Netherlands, 1986.

7 R. Davies. A temporal-logic approach to binding-time analysis. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27–30, 1996*, pages 184–195, 1996.

8 J. M. Davoren. On intuitionistic modal and tense logics and their classical companion logics: Topological semantics and bisimulations. *Annals of Pure and Applied Logic*, 161(3):349–367, 2009.

9 W. B. Ewald. Intuitionistic tense and modal logic. *The Journal of Symbolic Logic*, 51(1):166–179, 1986.

10 L. Fariñas del Cerro, A. Herzig, and E. Iraz Su. Epistemic equilibrium logic. In *IJCAI'15*, pages 2964–2970, Buenos Aires, Argentina, 2015. AAAI Press.

11 D. Fernández-Duque. The intuitionistic temporal logic of dynamical systems. *arXiv*, 1611.06929 [math.LO], 2016.

12 D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications, Volume 148 (Studies in Logic and the Foundations of Mathematics)*. North Holland, 1 edition, 2003.

13 D. Gabelaia, A. Kurucz, F. Wolter, and M. Zakharyaschev. Non-primitive recursive decidability of products of modal logics with expanding domains. *Annals of Pure and Applied Logic*, 142(1-3):245–268, 2006.

14 M. Gebser, C. Guziolowski, M. Ivanchev, T. Schaub, A. Siegel, S. Thiele, and P. Veber. Repair and prediction (under inconsistency) in large biological networks with answer set programming. In *KR'10*, 2010.

15 A. Heyting. *Die formalen Regeln der intuitionistischen Logik*. Sitzungsberichte der Preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse. Deütsche Akademie der Wissenschaften zu Berlin, Mathematisch-Naturwissenschaftliche Klasse, 1930.

16 D. Inclezan. An application of ASP to the field of second language acquisition. In *LPNMR'13*, pages 395–400, 2013.

17 N. Kamide and H. Wansing. Combining linear-time temporal logic with constructiveness and paraconsistency. *J. Applied Logic*, 8(1):33–61, 2010.

18 K. Kojima and A. Igarashi. Constructive linear-time temporal logic: Proof systems and Kripke semantics. *Information and Computation*, 209(12):1491 – 1503, 2011.

19 J. B. Kruskal. Well-quasi-ordering, the tree theorem, and vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225, 1960.

20 V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

21 V. Marek and M. Truszczyński. *Stable models and an alternative logic programming paradigm*, pages 169–181. Springer-Verlag, 1999.

22 G. Mints. *A Short Introduction to Intuitionistic Logic*. 2000.

23 I. Niemelä. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.

24 H. Nishimura. Semantical analysis of constructive PDL. *Publications of the Research Institute for Mathematical Sciences, Kyoto University*, 18:427–438, 1982.

25 M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry. An A-Prolog decision support system for the space shuttle. In *AAAI Spring Symposium*, 2001.

26 D. Pearce. A New Logical Characterisation of Stable Models and Answer Sets. In *Proc. of Non-Monotonic Extensions of Logic Programming (NMELP'96)*, pages 57–70, Bad Honnef, Germany, 1996.

27 G. Plotkin and C. Stirling. A framework for intuitionistic modal logics: Extended abstract. In *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, TARK'86, pages 399–406, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

28 A. K. Simpson. *The proof theory and semantics of intuitionistic modal logic*. PhD thesis, University of Edinburgh, UK, 1994.

A Proof of Lemma 30

Proof. Let us consider the stratified model $\mathcal{M} = (W, \preccurlyeq, S, V)$ obtained after the third phase. For all $k < i$ (where i has the value assigned at the end of this phase), W_k is a copy either of a normalized Σ-quasimodel or of a pointed normalized Σ-quasimodel. By Propositions 24 and 27, for all $k < i$, $|W_k| \leq Q_{s+3}^{2^{s+1}}$. We prove now that

$$i \leq 2E_{s+1}^{2^s} + sQ_{s+1}^{2^s} E_{s+3}^{2^{s+1}}.$$

After the first phase, by Proposition 24, we have $j \leq E_{s+1}^{2^s}$ and $|W_j| \leq Q_{s+1}^{2^s}$. Therefore, during the second phase, the current eventuality is defined at most $sQ_{s+1}^{2^s}$ times. Moreover, each time the current eventuality is undefined, by Proposition 27 we have that $i - \ell \leq E_{s+3}^{2^{s+1}}$ we have that $i - \ell \leq E_{n+3}^{2^{n+1}}$. Therefore, when the second phase terminates,

$$\ell - j \leq sQ_{s+1}^{2^s} E_{s+3}^{2^{s+1}}.$$

Finally, after the third phase, by Proposition 24, $i - \ell \leq E_{s+1}^{2^s}$. ◀

Decidable Logics with Associative Binary Modalities*

Joseph Boudou

IRIT – Toulouse University, Toulouse, France
joseph.boudou@irit.fr

Abstract

A new family of modal logics with an associative binary modality, called *counting logics* is proposed. These propositional logics allow to express finite cardinalities of sets and more generally to count the number of subsets satisfying some properties. We show that these logics can be seen both as specializations of the Boolean logic of bunched implications and as generalizations of the propositional dependence logic. Moreover, whereas most logics with an associative binary modality are undecidable, we prove that some counting logics are decidable, in particular the basic counting logic bCL. We conjecture that this interesting result is due to the valuation constraints in counting logics' semantics and prove that the logic corresponding to bCL without these constraints is undecidable. Finally, we give lower and upper bounds for the complexity of bCL's validity problem.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases modal logics, abstract separation logics, team semantics, resource logics, substructural logics

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.15

1 Introduction

Although most modal logics have only unary modalities, the concept of modality can easily be extended to any arity. In the present work, we are interested in binary modalities with a relational semantics: for an existential[1] binary modality $*$ there is a ternary relation R such that any formula of the form $\varphi * \psi$ is true at a state w if and only if there are two states x, y such that $R(w, x, y)$, φ is true at x and ψ is true at y. In many logics with such a binary modality, like separation logics [14], logics of bunched implications [13], interval logics [16] and ambient logics [5], the ternary relation R is interpreted as a decomposition of the current state into its constituents: $R(w, x, y)$ is read both as "the state w can be decomposed into the states x and y" and as "combining the states x and y produces the state w". In such a reading, it is usually expected that if a state w can be obtained by combining the state x with the combination of the states y and z then w can also be obtained by first combining x with y and then combining the resulting state with z. This requirement makes the binary modality associative. But it has been proved by Kurucz, Németi, Sain and Simon [8] that most logics with an associative binary modality are undecidable. Hence, many of the aforementioned

* This research was partially supported by ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02.

[1] Whereas for unary modality, the universal modality $\Box$ is generally considered as the principal modality and the existential modality $\Diamond$ is defined as its dual, it is generally the contrary for binary modalities. The reason is the close relation between existential binary modalities and tensors of substructural logics.

logics are undecidable, in particular the more general and abstract ones like the Boolean logic of bunched implications BBI [9, 3].

Despite this fact, a few specialized logics with an associative binary modality are decidable, like the propositional separation logic kSL0 [4], the propositional dependence logic [18] and Pandya's interval logic [12]. In the present work, we try to explain the reason why these logics are decidable. We observe that in each of the aforementioned decidable logics, the valuation of atomic formulas at any state w is not free but depends on the valuation at the components of w. Put differently, the properties of any compound state are determined by the properties of its atomic constituents. We conjecture that this feature is the main cause of the decidability of these logics. To justify this claim, we propose a new family of logics with associative binary modalities that have this property. We call them *counting logics*. We prove that some logics in this family are decidable, including the basic counting logic bCL. More precisely, we prove that bCL's validity problem is NEXPTIME-hard and in 4EXPTIME. In order to start to establish the limit of decidability for logics with associative binary modalities, we prove that the variant of bCL with free valuations is undecidable and that counting logics are more specific than the undecidable logic BBI and more general than the decidable propositional dependence logic. This latter result gives new insights into the relation between team logics and logic of separation, which was already noticed in [1].

Finally, beside the theoretical interest of counting logics, we believe that counting logic could have practical applications. Indeed, in many situations, the properties of compound elements are determined by the properties of their constituents. It could be the case, for instance, for resources and for sets of agents. Moreover, as their name suggests, counting logics allow to count: they can express that an element is composed of a given finite number of atomic constituents and that it can be decomposed into a given finite number of components satisfying a given property.

2 Language and Semantics

Given a countable infinite set Φ_0 of propositional variables, the basic language Φ of counting logics is defined inductively by the following grammar:

$$\varphi, \psi \coloneqq p \mid \neg\varphi \mid \top \mid (\varphi \wedge \psi) \mid \boldsymbol{I} \mid (\varphi * \psi)$$

where p is any propositional variable. The missing operators of classical propositional logic can be defined in the usual way. These classical operators are called *additive*. They are complemented with the *multiplicative* conjunction $*$ and its neutral element $\boldsymbol{I}$. Hence the language of counting logics is closely related to the languages of substructural logics like the Boolean logic of bunched implications BBI [13]. Intuitively, the formula $(\varphi * \psi)$ means that the current state can be decomposed into two complementary states, one satisfying φ, the other satisfying ψ. As usual, the parentheses may be omitted for clarity, the negation having the highest priority and the implication the lowest. For any formula φ, $\mathrm{SF}(\varphi)$, $\mathrm{PV}(\varphi)$ and $|\varphi|$ denotes, respectively, the set of subformulas of φ, the set of propositional variables occurring in φ and the cardinality of $\mathrm{SF}(\varphi)$. Finally, the following abbreviations are defined:

- The dual $\circledast$ of $*$ is defined by $\varphi \circledast \psi \doteq \neg(\neg\varphi * \neg\psi)$.
- The minimal cardinality predicate $\geq n$ is defined inductively by $\geq 0 \doteq \top$ and for all $n > 1$, $\geq n \doteq \neg\boldsymbol{I} * \geq(n-1)$.
- The cardinality predicate $=n$ is defined for all $n \in \mathbb{N}$ by $=n \doteq \geq n \wedge \neg\geq(n+1)$.
- The universal modality $\Box$ is defined by $\Box\varphi \doteq \bot \circledast \varphi$.

The extension $\Phi^{-\!\!*}$ of Φ with a multiplicative conditional will also be considered. The additional symbol $-\!\!*$, called the *magic wand*, is the residual of the multiplicative conjunction. Intuitively, the formula $(\varphi -\!\!* \psi)$ means that any state obtained by composing the current state with any state satisfying φ satisfies ψ. In fact, $\Phi^{-\!\!*}$ is exactly the language of the Boolean logic of bunched implications BBI.

A *subset space model* is a tuple $\mathcal{M} = (X, \mathcal{O}, V)$ where X is an arbitrary[2] set called the *universe*, $\mathcal{O}$ is a non-empty set of subsets of X and V is valuation function assigning a subset[3] of X to each propositional variable in Φ_0. We write V^{-1} for the function from X to the powerset of Φ_0, defined by $V^{-1}(x) \doteq \{p \in \Phi_0 \mid x \in V(p)\}$.

Formulas of Φ are interpreted at subsets in subset space models. We write $\mathcal{M}, S \vDash_C \varphi$ if the formula $\varphi \in \Phi$ is satisfied at the subset $S \in \mathcal{O}$ in the subset space model $\mathcal{M} = (X, \mathcal{O}, V)$. The relation $\vDash_C$ is defined inductively by:

$$
\begin{array}{ll}
\mathcal{M}, S \vDash_C p & \text{iff } S \cap V(p) \neq \emptyset \\
\mathcal{M}, S \vDash_C \top & \text{always} \\
\mathcal{M}, S \vDash_C \neg\varphi & \text{iff } \mathcal{M}, S \nvDash \varphi \\
\mathcal{M}, S \vDash_C \varphi \wedge \psi & \text{iff } \mathcal{M}, S \vDash_C \varphi \text{ and } \mathcal{M}, S \vDash_C \psi \\
\mathcal{M}, S \vDash_C \boldsymbol{I} & \text{iff } S = \emptyset \\
\mathcal{M}, S \vDash_C \varphi * \psi & \text{iff there is } S_1, S_2 \in \mathcal{O} \text{ such that} \\
 & \qquad S = S_1 \uplus S_2, \mathcal{M}, S_1 \vDash_C \varphi \text{ and } \mathcal{M}, S_2 \vDash_C \psi
\end{array}
$$

where $\uplus$ is the disjoint union of sets, i.e., the partial binary operator over sets such that $A = B \uplus C$ iff $B \cap C = \emptyset$ and $A = B \cup C$. This interpretation is extended to the language $\Phi^{-\!\!*}$ by the following additional rule:

$$
\begin{array}{ll}
\mathcal{M}, S \vDash_C \varphi -\!\!* \psi & \text{iff for all } S_1, S_2 \in \mathcal{O}, \text{ if } S_2 = S_1 \uplus S \text{ and } \mathcal{M}, S_1 \vDash_C \varphi \\
 & \qquad \text{then } \mathcal{M}, S_2 \vDash_C \psi
\end{array}
$$

A formula $\varphi \in \Phi$ is *valid* in a subset space model $\mathcal{M} = (X, \mathcal{O}, V)$, denoted by $\mathcal{M} \vDash_C \varphi$, iff $\mathcal{M}, S \vDash_C \varphi$ for all $S \in \mathcal{O}$. It is valid in a class $\mathcal{C}$ of subset space models, denoted by $\mathcal{C} \vDash_C \varphi$, iff for all models $\mathcal{M}$ in $\mathcal{C}$, $\mathcal{M} \vDash_C \varphi$.

A subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ may have some of the following properties:
Closure under inclusion $\mathcal{M}$ is closed under inclusion if for all $S \in \mathcal{O}$ and $S_1 \subseteq S$, $S_1 \in \mathcal{O}$.
Finiteness $\mathcal{M}$ is finite if X is finite.
Finite completeness $\mathcal{M}$ is finitely complete if $\mathcal{O}$ is the set of all finite subsets of X.
Atomicity $\mathcal{M}$ is atomic if for all $x \in X$ there is a unique $p \in \Phi_0$ such that $x \in V(p)$.
A class $\mathcal{C}$ of subset space models is said to have any of these properties if all the models in $\mathcal{C}$ have the property.

A *counting logic* is any logic which is obtained by interpreting the language Φ (or any extension of it) in a class of subset space models that are closed under inclusion. The basic counting logic bCL is the logic obtained by interpreting Φ in the class $\mathcal{C}_{\text{all}}$ of all subset space models that are closed under inclusion. The name of these logics is justified by the following proposition.

▶ **Proposition 1.** *For any subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ that is closed under inclusion, any $S \in \mathcal{O}$ and any $n \in \mathbb{N}$, $\mathcal{M}, S \vDash \geq n$ iff $|S| \geq n$.*

[2] Since formulas are evaluated at subsets, X may be empty if $\mathcal{O} = \{\emptyset\}$.
[3] This subset does not need to belong to $\mathcal{O}$.

Proof. The proof is by induction on n. The base case and the left-to-right direction of the inductive case are trivial and do not use the closure under inclusion property of $\mathcal{M}$. For the right-to-left direction, suppose $|S| \geq n > 0$. There is $x \in S$ and since $\mathcal{M}$ is closed under inclusion, $\{x\}$ and $S \setminus \{x\}$ belong to $\mathcal{O}$. The conclusion is straightforward. ◀

Actually, this proposition can be generalized to any formula. Given any formula φ, construct inductively the sequence $(\sqsupseteq_k^\varphi)_{k\in\mathbb{N}}$ such that $\sqsupseteq_0^\varphi \doteq \top$ and $\sqsupseteq_{k+1}^\varphi \doteq \varphi * \sqsupseteq_k^\varphi$ for all $k \in \mathbb{N}$. The previous proof can easily be adapted to show that for any subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ that is closed under inclusion and any subset $S \in \mathcal{O}$, $\mathcal{M}, S \vDash_C \sqsupseteq_k^\varphi$ if and only if there is at least k disjoint subsets of S satisfying φ in $\mathcal{M}$.

It can easily be checked that the multiplicative conjunction is associative and commutative. Moreover it is a binary modality in the sense that, for any subformulas φ, ψ and χ and any class $\mathcal{C}$ of subset space models we have: $\mathcal{C} \vDash_C (\varphi \circledast (\psi \to \chi)) \to (\varphi \circledast \psi) \to (\varphi \circledast \chi)$ and if $\mathcal{C} \vDash_C \varphi$ then $\mathcal{C} \vDash_C \psi \circledast \varphi$. Nevertheless, no non-trivial counting logics are normal modal logics because the inference rule of uniform substitution is not admissible, as shown by the following proposition.

▶ **Proposition 2.** *Let $\mathcal{C}$ be any class of subset space models that is closed under inclusion and such that there is a model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}$ with $|\mathcal{O}| > 1$. The inference rule of uniform substitution is not admissible in the counting logic L obtained by interpreting the language Φ in $\mathcal{C}$.*

Proof. We prove that the formula $(p * \neg \boldsymbol{I}) \to p$ is valid in L while $(\boldsymbol{I} * \neg \boldsymbol{I}) \wedge \neg \boldsymbol{I}$ is satisfiable in L. Suppose $\mathcal{M}, S \vDash p * \neg \boldsymbol{I}$ for some model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}$ and some $S \in \mathcal{O}$. There is a subset $S_1 \subset S$ such that $S_1 \cap V(p) \neq \emptyset$. Therefore, $S \cap V(p) \neq \emptyset$ and $\mathcal{M}, S \vDash p$. Now, let $\mathcal{M} = (X, \mathcal{O}, V)$ be a model in $\mathcal{C}$ such that $|\mathcal{O}| > 1$. There is a subset $S \in \mathcal{O}$ such that $S \neq \emptyset$, hence $\mathcal{M}, S \vDash \neg \boldsymbol{I}$. Moreover, since $\mathcal{M}$ is closed under inclusion, $\emptyset \in \mathcal{O}$. Since $S = \emptyset \uplus S$, $\mathcal{M}, S \vDash \boldsymbol{I} * \neg \boldsymbol{I}$. ◀

The non-admissibility of the uniform substitution is clearly due to the constraint on the "valuation" of subsets in subset space models. Broadly, the valuation of a subset is not free but is determined by the valuation of its constituents. Observing that most decidable logics with an associative binary modalities have such kind of constraints, like the propositional separation logic $k\mathsf{SL}0$ [14, 4] and the propositional dependence logic [18], we conjecture that this property is a key feature for a logic with an associative binary modality to be decidable.

3 Undecidability of the free disjoint union logic

We justify our claim that the constraint on the valuation of subsets in the semantics of counting logic is essential for the decidability of the basic counting logic. For that purpose, a new logic is defined that corresponds to the basic counting logic with free valuations. The logic is called the free disjoint union logic FDUL, and is proved to be undecidable.

A disjoint union model is a triple $\mathcal{M} = (\Omega, W, V)$ where Ω is a set, W a non-empty set of subsets of Ω that is closed under inclusion and V a valuation function assigning a subset of W to each propositional variable. Observe that the only difference with subset space models is that the valuation assigns subsets of W instead of subsets of Ω. The new satisfiability relation $\vDash_F$ between disjoint union models $\mathcal{M} = (\Omega, W, V)$, subsets $S \in W$ and formulas in Φ is defined by similar rules than for $\vDash_C$, except for the propositional variables in which case $\mathcal{M}, S \vDash_F p$ iff $S \in V(p)$. The free disjoint union logic FDUL is the logic obtained by interpreting Φ in the class of all disjoint union models. We first prove that FDUL does not have the finite model property with respect to these semantics.

▶ **Lemma 3.** *If* $\mathcal{M}, S \vDash_F p \wedge \Box(q \to {=}1) \wedge \Box(p \to (p * q))$ *then there is an infinite sequence* $x_1 x_2 \ldots$ *of pairwise distinct elements of* S *such that for all* $k \geq 1$, $\{x_k\} \in V(q)$ *and* $S \setminus \{x_1, \ldots, x_k\} \in V(p)$.

Proof. The infinite sequence $x_1 x_2 \ldots$ is constructed by induction on k. For $k = 1$, since $\mathcal{M}, S \vDash_F (p * q) \wedge \Box(q \to {=}1)$, there is $x_1 \in S$ such that $\{x_1\} \in V(q)$ and $S \setminus \{x_1\} \in V(p)$. For $k > 1$, let $S_k \doteq S \setminus \{x_1, \ldots, x_{k-1}\}$. By induction hypothesis, $S_k \in V(p)$. Therefore $\mathcal{M}, S_k \vDash_F (p * q) \wedge \Box(q \to {=}1)$ and the construction is similar as in the base case. ◀

To prove the undecidability of FDUL's satisfiability problem, the algebraic method proposed in [8] is difficult to apply because the underlying binary operator for the binary modality is partial. We use instead the following tiling problem [6]. A tiles set is a triple $(\mathcal{T}, \mathit{horz}, \mathit{vert})$ where $\mathcal{T}$ is a finite set whose elements are called tiles, and horz and vert are binary relations over $\mathcal{T}$. A tiling of $\mathbb{N} \times \mathbb{N}$ by the tiles set $(\mathcal{T}, \mathit{vert}, \mathit{horz})$ is a function $t : \mathbb{N} \times \mathbb{N} \longrightarrow \mathcal{T}$ such that for all $(x, y) \in \mathbb{N} \times \mathbb{N}$, $(t(x, y), t(x + 1, y)) \in \mathit{horz}$ and $(t(x, y), t(x, y + 1)) \in \mathit{vert}$. The $\mathbb{N} \times \mathbb{N}$ tiling problem consists in deciding whether there is a tiling of $\mathbb{N} \times \mathbb{N}$ by a given tiles set. This problem has been proved to be undecidable in [2].

Given a tiles set $(\mathcal{T}, \mathit{horz}, \mathit{vert})$, we construct a formula $\Psi \in \Phi$ such that Ψ is FDUL-satisfiable if and only if there is a tiling of $\mathbb{N} \times \mathbb{N}$ by $(\mathcal{T}, \mathit{horz}, \mathit{vert})$. First, we associate to each tile $i \in \mathcal{T}$ three propositional variables p_i, r_i and u_i. Intuitively, p_i, r_i and u_i state that the tile i is at the current position, on the right and above, respectively. Additionally, we use the propositional variables c, v and h, whose intuitive meaning will become clear shortly. All the aforementioned propositional variables must be pairwise distinct. The formula Ψ is defined as the conjunction of the following subformulas.

$$c \wedge \Box(v \vee h \to {=}1 \wedge (\neg v \vee \neg h)) \wedge \Box(c \to (v * c) \wedge (h * c)) \tag{1}$$

$$\Box\left(c \to \bigvee_{i \in \mathcal{T}} p_i\right) \tag{2}$$

$$\bigwedge_{i \in \mathcal{T}} \Box\left(p_i \to \bigwedge_{j \neq i} \neg p_j \wedge \bigvee_{j \in \mathit{horz}(i)} r_j \wedge \bigvee_{j \in \mathit{vert}(i)} u_j\right) \tag{3}$$

$$\bigwedge_{i \in \mathcal{T}} \Box\left(u_i \to (\neg v \circledast (c \wedge p_i)) \wedge \bigwedge_{j \neq i} \neg u_j\right) \tag{4}$$

$$\bigwedge_{i \in \mathcal{T}} \Box\left(r_i \to (\neg h \circledast (c \wedge p_i)) \wedge \bigwedge_{j \neq i} \neg r_j\right) \tag{5}$$

▶ **Lemma 4.** *There is a tiling of* $\mathbb{N} \times \mathbb{N}$ *by* $(\mathcal{T}, \mathit{horz}, \mathit{vert})$ *if and only if* Ψ *is FDUL-satisfiable.*

Proof. For the left-to-right direction, suppose there is a tiling t of $\mathbb{N} \times \mathbb{N}$ by $(\mathcal{T}, \mathit{horz}, \mathit{vert})$. Then for each cofinite subset S of $\mathbb{N}$, let x_S be the number of even numbers not in S and y_S the number of odd numbers not in S. The model $\mathcal{M} = (\Omega, W, V)$ is constructed such that $\Omega = \mathbb{N}$, $W = \mathcal{P}(\mathbb{N})$, $V(c)$ is the set of all cofinite subsets, $V(h)$ is the set of all singletons whose element is even, $V(v)$ is the set of all singletons whose element is odd, and for all $i \in \mathcal{T}$, a set $S \subseteq \mathbb{N}$ belongs to $V(p_i)$ (resp. r_i and u_i) iff S is cofinite and $t(x_S, y_S) = i$ (resp. $t(x_S + 1, y_S) = i$ and $t(x_S, y_S + 1) = i$). It can easily be checked that $\mathcal{M}, \mathbb{N} \vDash_F \Psi$. For instance for (4), suppose that $\mathcal{M}, S \vDash_F u_i$ and $S = S_1 \uplus S_2$. By definition, S is cofinite. Moreover, if $\mathcal{M}, S_1 \vDash_F v$ then S_1 is a singleton whose element is odd, S_2 is cofinite and $(x_{S_2}, y_{S_2}) = (x_S, y_S + 1)$, hence $\mathcal{M}, S_2 \vDash_F c \wedge p_i$.

For the right-to-left direction, suppose $\mathcal{M}, S \vDash_F \Psi$ for some $\mathcal{M} = (\Omega, W, V)$ and $S \in W$. Since S satisfies (1), by Lemma 3, there are two sequences $x_1 x_2 \ldots$ and $y_1 y_2 \ldots$ of distinct elements of S such that for all $k \geq 1$, $\{x_k\} \in V(h)$, $\{y_k\} \in V(v)$ and $S \setminus \{x_1, \ldots, x_k\} \in V(c)$. For all $(x, y) \in \mathbb{N} \times \mathbb{N}$, let $\mathcal{S}(x, y) \doteq S \setminus (\{x_k \mid k \leq x\} \cup \{y_k \mid k \leq y\})$. Using (4), it can easily be proved by induction on y that for all $(x, y) \in \mathbb{N} \times \mathbb{N}$, $\mathcal{S}(x, y) \in V(c)$. By (2) and (3), the function f from $V(c)$ to $\mathcal{T}$ can be defined such that $f(S') = i$ iff $S' \in V(p_i)$. It can easily be checked that the composition of f and $\mathcal{S}$ is a tiling of $\mathbb{N} \times \mathbb{N}$ by $(\mathcal{T}, \mathit{horz}, \mathit{vert})$. ◀

As a consequence, we have the following proposition.

▶ **Proposition 5.** *The satisfiability problem of FDUL is undecidable.*

4 Finite Model Property

We prove that the basic counting logic bCL has the finite model property and that its validity problem is decidable. We believe that the most important result is the finite model property since decidability can be deduced from it (see Section 7). Moreover, neither the free disjoint union logic FDUL from Section 3 nor the basic counting logic with magic wand $\mathsf{bCL}^{-\!\!*}$ has this property (with respect to the proposed semantics).

▶ **Lemma 6.** *For any subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ and any subset $S \in \mathcal{O}$, if $\mathcal{M}, S \vDash_C \top \mathrel{-\!\!*} \neg(\neg \boldsymbol{I} \mathrel{-\!\!*} \bot)$ then X is infinite.*

Proof. Suppose that $\mathcal{M}, S \vDash_C \top \mathrel{-\!\!*} \neg(\neg \boldsymbol{I} \mathrel{-\!\!*} \bot)$, $\mathcal{M} = (X, \mathcal{O}, V)$ and X is finite. The cardinality n of S belongs to $\mathbb{N}$. It can easily be proved by induction on k that for all $k \geq n$, there is a subset $S_k \in \mathcal{O}$ with cardinality k such that $S \subseteq S_k$. ◀

Notice that the previous lemma is weaker than Lemma 3 since it does not prove that $\mathsf{bCL}^{-\!\!*}$ does not have the finite-subset model property: we do not know whether there is a formula of $\Phi^{-\!\!*}$ which is $\mathsf{bCL}^{-\!\!*}$-satisfiable only at infinite subsets.

The results of this section are obtained by a faithful translation from bCL to the monadic second-order logic without functions MSO. The language Φ_{MSO} of this logic is defined from a set of term variables and a set of predicate variables by the following grammar:

$$\varphi, \psi \coloneqq P(x) \mid x = y \mid \neg\varphi \mid \top \mid (\varphi \wedge \psi) \mid (\forall x.\varphi) \mid (\forall P.\varphi)$$

where x and y are term variables and P a predicate variable. The missing boolean operator and the existential quantifiers are defined as usual. The formulas of Φ_{MSO} are evaluated by the relation $\vDash_{\mathsf{MSO}}$ at triples $(\mathfrak{D}, \mathfrak{t}, \mathfrak{p})$ where $\mathfrak{D}$ is a non-empty set, $\mathfrak{t}$ is a function assigning an element of $\mathfrak{D}$ to each term variable and $\mathfrak{p}$ is a function assigning a subset of $\mathfrak{D}$ to each predicate variable. The definition of $\vDash_{\mathsf{MSO}}$ is standard.

We now define the translation function τ from the language Φ of bCL to the language Φ_{MSO} of MSO. First, to each propositional variable $p \in \Phi_0$ is associated the predicate variable Q_p. The translation τ is also parametrized by a predicate variable and is defined inductively as follows:

$$\tau(P, p) \doteq (\exists x.P(x) \wedge Q_p(x)) \qquad \tau(P, \top) \doteq \top$$
$$\tau(P, \boldsymbol{I}) \doteq (\forall x.\neg P(x)) \qquad \tau(P, \neg\varphi) \doteq \neg\tau(P, \varphi)$$
$$\tau(P, \varphi \wedge \psi) \doteq \tau(P, \varphi) \wedge \tau(P, \psi)$$
$$\tau(P, \varphi * \psi) \doteq (\exists P_1 \exists P_2.\, (\forall x.P(x) \leftrightarrow (P_1(x) \vee P_2(x))) \wedge (\forall x.\neg P_1(x) \vee \neg P_2(x)) \wedge \tau(P_1, \varphi) \wedge \tau(P_2, \psi))$$

where the symbols P_1 and P_2 are chosen to be distinct from P.

▶ **Lemma 7.** *The formula φ is bCL-satisfiable if and only if $\tau(P,\varphi)$ is MSO-satisfiable.*

Proof. For the left-to-right direction, suppose $\mathcal{M}, S \vDash_C \varphi$ and let $\mathfrak{D} \doteq S$. It can easily be proved by structural induction on ψ that for any subformula ψ of φ, any subset $S' \subseteq S$, any term interpretation $\mathfrak{t}$ and any predicate interpretation $\mathfrak{p}$ such that $\mathfrak{p}(P) = S'$ and $\mathfrak{p}(Q_p) = V(p)$ for all $p \in \Phi_0$, $\mathcal{M}, S' \vDash_C \psi$ if and only if $\mathfrak{D}, \mathfrak{t}, \mathfrak{p} \vDash_{\mathsf{MSO}} \tau(P,\psi)$.

For the right-to-left direction, suppose $\mathfrak{D}, \mathfrak{t}, \mathfrak{p} \vDash_{\mathsf{MSO}} \tau(P,\varphi)$ and let $\mathcal{M} \doteq (X, \mathcal{O}, V)$ such that $X = \mathfrak{p}(P)$, $\mathcal{O} = \mathcal{P}(\mathfrak{p}(P))$ and $V(p) = \mathfrak{p}(P) \cap \mathfrak{p}(Q_p)$ for all $p \in \Phi_0$. Clearly, $\mathcal{M}$ is a subset space model closed under inclusion. It can easily be proved by structural induction on ψ that for any subformula ψ of φ and any predicate interpretation $\mathfrak{p}'$ such that $\mathfrak{p}'(P) \subseteq \mathfrak{p}(P)$ and $\mathfrak{p}'(Q_p) = \mathfrak{p}(Q_p)$ for all $p \in \Phi_0$, $\mathfrak{D}, \mathfrak{t}, \mathfrak{p}' \vDash_{\mathsf{MSO}} \tau(P,\psi)$ if and only if $\mathcal{M}, \mathfrak{p}'(P) \vDash_C \psi$. ◀

Löwenheim [10] proved that MSO is decidable and has the finite model property. Hence, we have the following proposition.

▶ **Proposition 8.** *bCL has the finite model property and its validity problem is decidable.*

5 Comparison with BBI

The Boolean logic of bunched implications BBI [13] has been devised to reason about resources. This logic is usually considered as the foundation of separation logics [14] and other logics with a separative multiplicative conjunction [5]. The language of BBI is $\Phi^{-\!\!*}$ and we prove in this section that $\mathsf{bCL}^{-\!\!*}$ can be seen as BBI interpreted in a restricted class of BBI models. By the results in [8], the magic wand-free fragment of BBI is already undecidable, hence the present section gives new insight into the decidability of separation logics.

A *commutative partial monoid* is a triple $(M, \circ, \mathbf{e})$ where M is a set, $\circ$ a partial function from $M \times M$ to M and $\mathbf{e} \in M$, satisfying for all $a, b, c \in M$:

$$\mathbf{e} \circ a\!\downarrow \text{ and } \mathbf{e} \circ a = a \qquad \text{(identity)}$$

$$\text{if } a \circ b\!\downarrow \text{ then } b \circ a\!\downarrow \text{ and } a \circ b = b \circ a \qquad \text{(commutativity)}$$

$$\text{if } a \circ (b \circ c)\!\downarrow \text{ then } (a \circ b) \circ c\!\downarrow \text{ and } a \circ (b \circ c) = (a \circ b) \circ c \qquad \text{(associativity)}$$

where $a \circ b\!\downarrow$ denotes that $\circ$ is defined at (a, b). Notice that, for instance, $a \circ (b \circ c)\!\downarrow$ implies $b \circ c\!\downarrow$. A BBI model is a tuple $\mathcal{M} = (M, \circ, \mathbf{e}, V)$ where $(M, \circ, \mathbf{e})$ is a commutative partial monoid and V is a valuation function assigning a subset of M to each propositional variable. Formula in $\Phi^{-\!\!*}$ are interpreted in BBI models by the relation $\vDash_B$ defined by:

$$\begin{aligned}
&\mathcal{M}, a \vDash_B p && \text{iff } a \in V(p)\\
&\mathcal{M}, a \vDash_B \boldsymbol{I} && \text{iff } a = \mathbf{e}\\
&\mathcal{M}, a \vDash_B \varphi * \psi && \text{iff there is } b, c \in M \text{ s.t. } b \circ c\!\downarrow,\ a = b \circ c,\ \mathcal{M}, b \vDash_B \varphi \text{ and } \mathcal{M}, c \vDash_B \psi\\
&\mathcal{M}, a \vDash_B \varphi \mathrel{-\!\!*} \psi && \text{iff for all } b, c \in M \text{ if } a \circ b\!\downarrow,\ a \circ b = c \text{ and } \mathcal{M}, b \vDash_B \varphi \text{ then } \mathcal{M}, c \vDash_B \psi
\end{aligned}$$

the omitted rules for the Boolean constructs being classical. The logic obtained by interpreting the language $\Phi^{-\!\!*}$ in the class of all BBI models is BBI[4].

We now describe the class of BBI models which corresponds to $\mathsf{bCL}^{-\!\!*}$. Let $(M, \circ, \mathbf{e})$ be a commutative partial monoid. The binary relation $\preccurlyeq$ on M is defined such that for all $a, b \in M$, $a \preccurlyeq b$ iff $b = a \circ c$ for some $c \in M$. An element $a \in M$ is an *atom*

[4] More precisely, this logic is usually called $\mathsf{BBI}_{\mathsf{PD}}$, for instance in [9].

if $a \neq \mathbf{e}$ and for all b, if $b \preccurlyeq a$ then $b \in \{\mathbf{e}, a\}$. For any element $b \in M$, we define $\mathrm{At}(b) \doteq \{a \in M \mid a \text{ is an atom and } a \preccurlyeq b\}$. We define the class $\mathcal{M}_{\mathsf{CL}}$ of all the commutative partial monoids $(M, \circ, \mathbf{e})$ satisfying the following properties:

Atomicity for all $b \in M \setminus \{\mathbf{e}\}$, there is an atom $a \in M$ such that $a \preccurlyeq b$;

Disjointness for all $a \in M$, if $a \circ a{\downarrow}$ then $a = \mathbf{e}$;

Weak cross-split for all $a, b, c, d \in M$, if $a \circ b = c \circ d$ and $a \neq \mathbf{e}$ then there is $s \in M$ such that $s \neq \mathbf{e}$, $s \preccurlyeq a$ and $s \preccurlyeq c$ or $s \preccurlyeq d$;

Bounded completeness every set $S \subseteq M$ that has an upper bound with respect to $\preccurlyeq$ has a least upper bound.

For the last property to make sense it has to be observed that the relation $\preccurlyeq$ is a partial order on any disjoint commutative partial monoid. The class $\mathcal{C}_{\mathsf{CL}}$ of BBI models is defined as all the BBI models $\mathcal{M} = (M, \circ, \mathbf{e}, V)$ such that $(M, \circ, \mathbf{e})$ is in $\mathcal{M}_{\mathsf{CL}}$ and for all $p \in \Phi_0$ and all $b \in M$ that is not an atom, $b \in V(p)$ iff there is an atom $a \in M$ such that $a \preccurlyeq b$ and $a \in V(p)$.

▶ **Lemma 9.** *Every commutative partial monoid $(M, \circ, \mathbf{e})$ in $\mathcal{M}_{\mathsf{CL}}$ is atomistic, i.e., any element $a \in M$ is the least upper bound of $\mathrm{At}(a)$.*

Proof. By definition, a is an upper bound of $\mathrm{At}(a)$ and since $(M, \circ, \mathbf{e})$ is bounded complete, there is a least upper bound b of $\mathrm{At}(a)$. Let us suppose that $b \neq a$. There must exists $c \in M$, different from $\mathbf{e}$, such that $a = b \circ c$. By atomicity, there is an atom $d \in M$ such that $d \preccurlyeq c$. Obviously $d \in \mathrm{At}(a)$ and by associativity and commutativity, $d \circ d{\downarrow}$. By disjointness, $d = \mathbf{e}$ which is not possible. ◀

▶ **Lemma 10.** *For any commutative partial monoid $(M, \circ, \mathbf{e})$ in $\mathcal{M}_{\mathsf{CL}}$ and any $a, b, c \in M$, $a = b \circ c$ iff $\mathrm{At}(a) = \mathrm{At}(b) \uplus \mathrm{At}(c)$.*

Proof. For the left-to-right direction, suppose that $a = b \circ c$. By the disjointness property, $\mathrm{At}(b) \cap \mathrm{At}(c) = \emptyset$. By transitivity of $\preccurlyeq$, $\mathrm{At}(b) \uplus \mathrm{At}(c) \subseteq \mathrm{At}(a)$. By the weak cross-split property, for any atom $d \in \mathrm{At}(a)$, $d \preccurlyeq b$ or $d \preccurlyeq c$.

For the right-to-left direction, suppose $\mathrm{At}(a) = \mathrm{At}(b) \uplus \mathrm{At}(c)$. By Lemma 9 on b, there is $c' \in M$ such that $a = b \circ c'$. For all $d \in \mathrm{At}(c')$, $d \in \mathrm{At}(a)$ and by disjointness $d \notin \mathrm{At}(b)$, hence $\mathrm{At}(c') \subseteq \mathrm{At}(c)$. For all $d \in \mathrm{At}(c)$, $d \preccurlyeq b \circ c'$ and by the weak cross-split property $d \preccurlyeq b$ or $d \preccurlyeq c'$, hence $\mathrm{At}(c) \subseteq \mathrm{At}(c)$. Therefore $\mathrm{At}(c) = \mathrm{At}(c')$ and by Lemma 9, $c' = c$. ◀

▶ **Proposition 11.** *The logic obtained by interpreting $\Phi^{-\!\!*}$ in $\mathcal{C}_{\mathsf{CL}}$ is $\mathsf{bCL}^{-\!\!*}$.*

Proof. It can easily be checked that any formula that is satisfiable in a subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ that is closed under inclusion, is satisfiable in the BBI model $(\mathcal{O}, \uplus, \emptyset, V')$ where $S \in V'(p)$ iff $\mathcal{M}, S \vDash_C p$, and that this BBI model belongs to $\mathcal{C}_{\mathsf{CL}}$. For the other direction, from any BBI model $\mathcal{M} = (M, \circ, \mathbf{e}, V)$ in $\mathcal{C}_{\mathsf{CL}}$ construct the subset space model $\mathcal{M}' \doteq (X, \mathcal{O}, V')$ where X is the set of all atoms in M, $\mathcal{O}$ is the set of all subset of atoms that has an upper bound by $\preccurlyeq$ and V' is the reduction of V to X. It can easily be checked that $\mathcal{M}'$ is closed under inclusion and satisfies the same formulas than $\mathcal{M}$. ◀

6 Comparison with the Propositional Dependence logic

The team semantics has been devised by Hodges [7] to provide a compositional semantics for the independence-friendly logic. The main idea is to evaluate formulas at *sets* of valuations, called teams, instead of at single valuations. Following this idea, new propositional logics, called propositional team logics, has been devised and studied recently [17, 18, 19]. Observing

that, by identifying each element of a universe to a valuation, counting logics can be seen as generalizations of propositional team logics, we prove in this section that the *propositional downward closed team logic* PD [18], also called propositional dependence logic, can be embedded in a large class of counting logics.

We first recall the syntax and semantics of PD. Given a set Φ_0 of propositional variables, the language Φ_{PD} is defined inductively by:

$$\varphi, \psi \coloneqq p \mid \neg p \mid \bot \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \otimes \psi) \mid =(q_1, \dots, q_n, p)$$

where $p, q_1, \dots, q_n$ are propositional variables. The symbol $\otimes$ is called the *tensor* and formulas of the form $=(q_1, \dots, q_n, p)$ *dependence atoms*. Notice that negation is allowed only in front of propositional variables. A *valuation* is a subset $v \subseteq \Phi_0$ of propositional variables and a *team* T is a subset of valuations. Formulas in Φ_{PD} are evaluated at teams by the relation $\vDash_{\mathsf{PD}}$ defined inductively by:

$$\begin{aligned}
&T \vDash_{\mathsf{PD}} p && \text{iff for all } v \in T, p \in v \\
&T \vDash_{\mathsf{PD}} \neg p && \text{iff for all } v \in T, p \notin v \\
&T \vDash_{\mathsf{PD}} \bot && \text{iff } T = \emptyset \\
&T \vDash_{\mathsf{PD}} \varphi \wedge \psi && \text{iff } T \vDash_{\mathsf{PD}} \varphi \text{ and } T \vDash_{\mathsf{PD}} \psi \\
&T \vDash_{\mathsf{PD}} \varphi \vee \psi && \text{iff } T \vDash_{\mathsf{PD}} \varphi \text{ or } T \vDash_{\mathsf{PD}} \psi \\
&T \vDash_{\mathsf{PD}} \varphi \otimes \psi && \text{iff there is } T_1, T_2 \text{ such that } T = T_1 \cup T_2, T_1 \vDash_{\mathsf{PD}} \varphi \text{ and } T_2 \vDash_{\mathsf{PD}} \psi \\
&T \vDash_{\mathsf{PD}} =(q_1, \dots, q_n, p) && \text{iff for all } v_1, v_2 \in T, \text{ if } v_1 \cap \{q_1, \dots, q_n\} = v_2 \cap \{q_1, \dots, q_n\} \\
& && \qquad \text{then } v_1 \cap \{p\} = v_2 \cap \{p\}
\end{aligned}$$

A formula $\varphi \in \Phi_{\mathsf{PD}}$ is PD-valid, denoted by $\vDash_{\mathsf{PD}} \varphi$, if $T \vDash_{\mathsf{PD}} \varphi$ for all teams T. The propositional dependence logic PD is obtained by interpreting Φ_{PD} in the class of all teams over Φ_0. An important property of this logic is the *downward closure*: if $T \vDash_{\mathsf{PD}} \varphi$ then for any subset $T' \subseteq T$, $T' \vDash_{\mathsf{PD}} \varphi$ (see for instance [18]).

We prove that PD can be embedded in any counting logic obtained by interpreting Φ (or any of its extensions) in any class $\mathcal{C}$ of subset space models that are closed under inclusion and that satisfies the following *comprehensive condition*: for all $P \subseteq \Phi_0$, there is a subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}$ and a subset $S \in \mathcal{O}$ such that for any valuation $v \subseteq P$, there is $x \in S$ such that $v = V^{-1}(x)$. Remark that $\mathcal{C}_{\text{all}}$ satisfies the comprehensive condition. The translation τ from Φ_{PD} to Φ is defined inductively as follows:

$$\begin{aligned}
\tau(p) &\doteq \Box(\boldsymbol{I} \vee p) & \tau(\varphi \wedge \psi) &\doteq \tau(\varphi) \wedge \tau(\psi) \\
\tau(\neg p) &\doteq \neg p & \tau(\varphi \vee \psi) &\doteq \tau(\varphi) \vee \tau(\psi) \\
\tau(\bot) &\doteq \boldsymbol{I} & \tau(\varphi \otimes \psi) &\doteq \tau(\varphi) * \tau(\psi)
\end{aligned}$$

$$\tau(=(q_1, \dots, q_n, p)) \doteq \Box\left(=2 \to \left(\bigwedge_{i \in 1..n} \text{same}(q_i)\right) \to \text{same}(p)\right)$$

where same is defined by $\text{same}(p) \doteq (p * p) \vee ((\neg \boldsymbol{I} \wedge \neg p) * (\neg \boldsymbol{I} \wedge \neg p))$ for all propositional variables p. This translation is clearly polynomial. To prove that it preserves validity (Proposition 14), the following lemmas are needed. The proof of Lemma 12 is straightforward and Lemma 13 has already been proved, for instance in [17].

▶ **Lemma 12.** *For any subset space model $\mathcal{M} = (X, \mathcal{O}, V)$, any $\{x, y\} \in \mathcal{O}$ and any propositional variable $p \in \Phi_0$, $\mathcal{M}, \{x, y\} \vDash_C \text{same}(p)$ iff $\{x, y\} \subseteq V(p)$ or $\{x, y\} \cap V(p) = \emptyset$.*

▶ **Lemma 13.** *For any formula $\varphi \in \Phi$, let $T_\varphi \doteq \mathcal{P}\left(PV(\varphi)\right)$. Then φ is PD-valid iff $T_\varphi \vDash_{PD} \varphi$.*

▶ **Proposition 14.** *For any class $\mathcal{C}$ of subset space models that are closed under inclusion and any formula $\varphi \in \Phi_{PD}$, if $\mathcal{C}$ fulfils the comprehensive condition then $\vDash_{PD} \varphi$ iff $\mathcal{C} \vDash_C \tau(\varphi)$.*

Proof. For the left-to-right direction, suppose that $\vDash_{\mathsf{PD}} \varphi$. Let $\mathcal{M} = (X, \mathcal{O}, V)$ be a subset space model in $\mathcal{C}$. For all $S \in \mathcal{O}$, let $T(S) \doteq \left\{v \subseteq \mathrm{PV}(\varphi) \;\middle|\; \text{there is } x \in S,\ V^{-1}(x) = v\right\}$. We prove by structural induction on ψ that for all $\psi \in \mathrm{SF}(\varphi)$ and all $S \in \mathcal{O}$, if $T(S) \vDash_{\mathsf{PD}} \psi$ then $\mathcal{M}, S \vDash_C \tau(\psi)$. We detail only some cases, the missing ones being either similar or straightforward. For propositional variables, suppose that $\mathcal{M}, S \nvDash_C \Box(\boldsymbol{I} \vee p)$. Then $\mathcal{M}, S \vDash_C \top * (\neg\boldsymbol{I} \wedge \neg p)$. Hence there is a subset $S_2 \subseteq S$ such that $\mathcal{M}, S_2 \vDash_C \neg\boldsymbol{I} \wedge \neg p$. Therefore there is $x \in S$ such that $x \notin V(p)$. By definition, there is $v \in T(S)$ such that $p \notin v$. We have proved that $T(S) \nvDash_{\mathsf{PD}} p$. For tensor products, let us suppose that $T(S) \vDash_{\mathsf{PD}} \psi_1 \otimes \psi_2$. There are T_1, T_2 such that $T(S) = T_1 \cup T_2$, $T_1 \vDash_{\mathsf{PD}} \psi_1$ and $T_2 \vDash_{\mathsf{PD}} \psi_2$. By the downward closure property, $T_2 \setminus T_1 \vDash_{\mathsf{PD}} \psi_2$ too. Let $S_1 \doteq \left\{x \in S \;\middle|\; V^{-1}(x) \in T_1\right\}$ and $S_2 \doteq \left\{x \in S \;\middle|\; V^{-1}(x) \in T_2 \setminus T_1\right\}$. Clearly $S = S_1 \uplus S_2$ and by the induction hypothesis, $\mathcal{M}, S \vDash_C \tau(\psi_1) * \tau(\psi_2)$. For dependence atoms, suppose that $\mathcal{M}, S \nvDash_C \tau\left(=(q_1, \ldots, q_n, p)\right)$. There is $\{x, y\} \subseteq S$ such that $x \neq y$ and $\mathcal{M}, \{x, y\} \vDash_C \left(\bigwedge_{i \in 1..n} \mathrm{same}(q_i)\right) \wedge \neg\mathrm{same}(p)$. By Lemma 12, $V^{-1}(x) \cap \{q_1, \ldots, q_n\} = V^{-1}(y) \cap \{q_1, \ldots, q_n\}$ and $V^{-1}(x) \cap \{p\} \neq V^{-1}(y) \cap \{p\}$. Therefore $T(S) \nvDash_{\mathsf{PD}} =(q_1, \ldots, q_n, p)$.

For the right-to-left direction, suppose that $\mathcal{C} \vDash_C \tau(\varphi)$. By the comprehensive condition, there is $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}$ and $S \in \mathcal{O}$ such that $T_\varphi \subseteq T(S)$ where T_φ is defined as in Lemma 13 and $T(S)$ as in the previous direction. By Lemma 13 and the downward closure property, it is sufficient to prove that for all $\psi \in \mathrm{SF}(\varphi)$ and all $S' \subseteq S$, if $\mathcal{M}, S' \vDash_C \tau(\psi)$ then $T(S') \vDash_{\mathsf{PD}} \psi$. The proof is by structural induction on ψ. We detail only some cases, the missing ones being either similar or straightforward. For propositional variables, suppose that $T(S') \nvDash_{\mathsf{PD}} p$. There is $v \in T(S')$ such that $p \notin v$. By definition, there is $x \in S'$ such that $x \notin V(p)$. Therefore $\mathcal{M}, \{x\} \vDash \neg\boldsymbol{I} \wedge \neg p$ and $\mathcal{M}, S' \nvDash_C \Box(\boldsymbol{I} \vee p)$. For tensor products, suppose that $\mathcal{M}, S' \vDash_C \tau(\psi_1) * \tau(\psi_2)$. There are $S_1, S_2 \in \mathcal{O}$ such that $S' = S_1 \uplus S_2$, $\mathcal{M}, S_1 \vDash_C \tau(\psi_1)$ and $\mathcal{M}, S_2 \vDash_C \tau(\psi_2)$. Obviously, $T(S_1) \cup T(S_2) = T(S_1 \cup S_2)$. Therefore, by induction hypothesis, $T(S') \vDash_{\mathsf{PD}} \psi_1 \otimes \psi_2$. For dependence atoms, suppose that $T(S') \nvDash_{\mathsf{PD}} =(q_1, \ldots, q_n, p)$. There are $x, y \in S'$ such that $V^{-1}(x) \cap \{q_1, \ldots, q_n\} = V^{-1}(y) \cap \{q_1, \ldots, q_n\}$ and $V^{-1}(x) \cap \{p\} \neq V^{-1}(y) \cap \{p\}$. By Lemma 12, $\mathcal{M}, \{x, y\} \vDash_C \left(\bigwedge_{i \in 1..n} \mathrm{same}(q_i)\right) \wedge \neg\mathrm{same}(p)$. Therefore $\mathcal{M}, S' \nvDash_C \tau\left(=(q_1, \ldots, q_n, p)\right)$. ◀

Since the validity problem of PD has been proved in [17] to be NEXPTIME-complete, and the translation is polynomial, we have the following corollary. It implies that bCL's validity problem is NEXPTIME-hard, because $\mathcal{C}_{\mathrm{all}}$ satisfies the comprehensive condition.

▶ **Corollary 15.** *The validity problem of any counting logic obtained by interpreting Φ (or any of its extension) in a class of subset space model that is closed under inclusion and that satisfies the comprehensive condition is NEXPTIME-hard.*

7 Complexity upper bounds

We proved in Section 4 that the basic counting logic bCL is decidable, and in the previous section that its validity problem is NEXPTIME-hard. We now establish a 4EXPTIME upper bound. For that matter, we consider two new counting logics. The finitely complete counting logic $\mathsf{CL}^{*}_{\mathrm{comp}}$ is the logic obtained by interpreting the language Φ^{*} in the class $\mathcal{C}_{\mathrm{comp}}$ of all finitely complete subset models. The finitely complete atomic counting logic $\mathsf{CL}^{*}_{\mathrm{atom}}$ is the

logic obtained by interpreting the language $\Phi^{-\!\!*}$ in the class $\mathcal{C}_{\text{atom}}$ of all subset models that are finitely complete and atomic. We first prove that $\mathsf{CL}^{-\!\!*}_{\text{atom}}$'s satisfiability problem is in 3EXPTIME by a reduction to the satisfiability of Presburger arithmetic. Then we provide a faithful but exponential translation from $\mathsf{CL}^{-\!\!*}_{\text{comp}}$ to $\mathsf{CL}^{-\!\!*}_{\text{atom}}$. Finally we prove that bCL is the magic wand-free fragment of $\mathsf{CL}^{-\!\!*}_{\text{comp}}$.

7.1 Complexity of the satisfiability problem of $\mathsf{CL}^{-\!\!*}_{\text{atom}}$

We provide a polynomial reduction of $\mathsf{CL}^{-\!\!*}_{\text{atom}}$'s satisfiability problem to the satisfiability problem of Presburger arithmetic Presburger arithmetic is the first-order theory of the natural numbers $\mathbb{N}$ with addition $+$. The set of quantified variables of Presburger arithmetic is denoted by $\mathbb{V}$. Formulas of this theory are called *constraints*. A Presburger assignment[5] is a function $\mathcal{A}$ from $\mathbb{V}$ to $\mathbb{N}$. We write $\mathcal{A} \vDash_{\mathsf{PA}} C$ to denote that the constraint C is satisfied by the assignment $\mathcal{A}$.

Given a formula $\varphi_0 \in \Phi^{-\!\!*}$, we construct a constraint that is satisfiable in Presburger arithmetic if and only if φ_0 is $\mathsf{CL}^{-\!\!*}_{\text{atom}}$-satisfiable. For that purpose, we use *vectors* which are ordered finite sets, i.e., finite sequences without repetitions (each component of a vector occurs exactly once). Vectors are distinguished by an arrow accent. We write v_i to denote the i^{th} component of $\overrightarrow{v}$. We may abusively consider a vector to be the set of its components and write for instance $u \in \overrightarrow{v}$. In this section, it is assumed that all vectors have length $n \doteq |\text{PV}(\varphi_0)| + 1$. From the formula φ_0, a vector $\overrightarrow{p}$ of propositional variables is defined such that $\{p_1, \ldots, p_{n-1}\} = \text{PV}(\varphi_0)$ and $p_n \notin \text{PV}(\varphi_0)$. Moreover, two vectors $\overrightarrow{r}$ and $\overrightarrow{s}$ of Presburger arithmetic's variables are selected such that $\overrightarrow{r} \cap \overrightarrow{s} = \emptyset$.

A model $\mathcal{M} = (X, \mathcal{O}, V)$ is *flat* on a subset $P \subseteq \Phi_0$ of propositional variables if $V(p) = \emptyset$ for all $p \in P$.

▶ **Lemma 16.** *If φ_0 is $\mathsf{CL}^{-\!\!*}_{atom}$-satisfiable then it is satisfiable in a model from $\mathcal{C}_{atom}$ that is flat on $\Phi_0 \setminus \overrightarrow{p}$.*

Proof. Suppose that $\mathcal{M}, S \vDash_C \varphi$. Construct $\mathcal{M}' \doteq (X, \mathcal{O}, V')$ such that $V'(p) = V(p)$ for all $p \in \text{PV}(\varphi_0)$, $V'(p_n) = \bigcup_{p \notin \text{PV}(\varphi_0)} V(p)$ and $V'(p) = \emptyset$ for all $p \notin \overrightarrow{p}$. It can easily be proved by structural induction on ψ that for all $\psi \in \text{SF}(\varphi_0)$ and all $S' \in \mathcal{O}$, $\mathcal{M}', S' \vDash_C \psi$ iff $\mathcal{M}, S' \vDash_C \psi$. ◀

A pair $(\mathcal{M}, S)$ composed of a subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}_{\text{atom}}$ and a subset $S \in \mathcal{O}$ *is in correspondence with* a Presburger arithmetic assignment $\mathcal{A}$ by a vector $\overrightarrow{x}$ of Presburger arithmetic variables, denoted by $\mathcal{M}, S \xrightarrow{\overrightarrow{x}} \mathcal{A}$, if all the following conditions hold for all $i \in 1..n$:

- $\mathcal{M}$ is flat on $\Phi_0 \setminus \overrightarrow{p}$;
- if $\mathcal{A}(r_i) = 0$ then $|V(p_i)| = \mathcal{A}(s_i)$;
- if $\mathcal{A}(r_i) > 0$ then $|V(p_i)|$ is infinite;
- $|V(p_i) \cap S| = \mathcal{A}(x_i)$.

Clearly, for any model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}_{\text{atom}}$ that is flat on $\Phi_0 \setminus \overrightarrow{p}$, any subset $S \in \mathcal{O}$ and any vector $\overrightarrow{x}$ of Presburger arithmetic disjoint from $\overrightarrow{r}$ and $\overrightarrow{s}$ (i.e., $\overrightarrow{x} \cap \overrightarrow{r} = \overrightarrow{x} \cap \overrightarrow{s} = \emptyset$), there is a Presburger arithmetic assignment $\mathcal{A}$ such that $\mathcal{M}, S \xrightarrow{\overrightarrow{x}} \mathcal{A}$.

[5] For the sake of simplicity, we only consider here the standard interpretation of Presburger arithmetic over natural numbers.

As a first component of the translation of φ_0, the constraint $\Psi(\overrightarrow{x})$ is defined for all vectors $\overrightarrow{x}$ of Presburger arithmetic variables that is disjoint from $\overrightarrow{r}$ and $\overrightarrow{s}$, by:

$$\Psi(\overrightarrow{x}) \doteq \bigwedge_{i \in 1..n} (r_i = 0 \to x_i \leq s_i)$$

As stated by the following lemma, this formula ensures that there is a pair $(\mathcal{M}, S)$ in correspondence with any assignment satisfying it.

▶ **Lemma 17.** *For any vector $\overrightarrow{x}$ of Presburger arithmetic variables that is disjoint from $\overrightarrow{r}$ and $\overrightarrow{s}$ and for any Presburger arithmetic assignment $\mathcal{A}$, $\mathcal{A} \vDash_{PA} \Psi(\overrightarrow{x})$ if and only if there is a subset space model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}_{atom}$ and a subset $S \in \mathcal{O}$ such that $\mathcal{M}, S \xrightarrow{\overrightarrow{x}} \mathcal{A}$.*

Proof. For the left-to-right direction, suppose $\mathcal{A} \vDash_{\mathsf{PA}} \Psi(\overrightarrow{x})$. Construct $\mathcal{M} = (X, \mathcal{O}, V)$ and S such that $X = \{(i,j) \in \{1\,..\,n\} \times \mathbb{N} \mid \text{if } \mathcal{A}(r_i) = 0 \text{ then } j < \mathcal{A}(s_i)\}$, $\mathcal{O} = \mathcal{P}(X)$, $V(p_i) = \{(i',j) \in X \mid i' = i\}$ for all $i \in 1\,..\,n$, $V(q) = \emptyset$ for all $q \notin \overrightarrow{p}$ and $S = \{(i,j) \in X \mid j < \mathcal{A}(x_i)\}$. It can easily be checked that $\mathcal{M}$ belongs to $\mathcal{C}_{\text{atom}}$ and $\mathcal{M}, S \xrightarrow{\overrightarrow{x}} \mathcal{A}$. The right-to-left direction is straightforward. ◀

The second component of the translation is the function θ assigning a constraint to any pair $(\overrightarrow{x}, \psi)$ composed of a vector $\overrightarrow{x}$ of Presburger arithmetic variables that is disjoint from $\overrightarrow{r}$ and $\overrightarrow{s}$ and a subformula ψ of φ_0. This function is defined inductively as follows:

$$\theta(\overrightarrow{x}, p_i) \doteq x_i > 0 \qquad \theta(\overrightarrow{x}, \top) \doteq \top$$

$$\theta(\overrightarrow{x}, \boldsymbol{I}) \doteq \bigwedge_{i \in 1..n} x_i = 0 \qquad \theta(\overrightarrow{x}, \neg\varphi) \doteq \neg\theta(\overrightarrow{x}, \varphi)$$

$$\theta(\overrightarrow{x}, \varphi \wedge \psi) \doteq \theta(\overrightarrow{x}, \varphi) \wedge \theta(\overrightarrow{x}, \psi)$$

$$\theta(\overrightarrow{x}, \varphi * \psi) \doteq (\exists \overrightarrow{y}, \overrightarrow{z}.\ \bigwedge_{i \in 1..n} x_i = y_i + z_i \ \wedge\ \Psi(\overrightarrow{y}) \wedge \theta(\overrightarrow{y}, \varphi) \wedge \Psi(\overrightarrow{z}) \wedge \theta(\overrightarrow{z}, \psi))$$

$$\theta(\overrightarrow{x}, \varphi \mathrel{-\!\!*} \psi) \doteq (\forall \overrightarrow{y}, \overrightarrow{z}.\ \bigwedge_{i \in 1..n} z_i = x_i + y_i \ \to\ \Psi(\overrightarrow{y}) \to \theta(\overrightarrow{y}, \varphi) \to \Psi(\overrightarrow{z}) \to \theta(\overrightarrow{z}, \psi))$$

where $\overrightarrow{y}$ and $\overrightarrow{z}$ are chosen such that $\overrightarrow{r}$, $\overrightarrow{s}$, $\overrightarrow{x}$, $\overrightarrow{y}$ and $\overrightarrow{z}$ are pairwise disjoint.

▶ **Lemma 18.** *For any subformula ψ of φ_0, any Presburger arithmetic assignment $\mathcal{A}$, any vector $\overrightarrow{x}$ of Presburger arithmetic variables that is disjoint from $\overrightarrow{r}$ and $\overrightarrow{s}$, any model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}_{atom}$ and any subset $S \in \mathcal{O}$ such that $\mathcal{M}, S \xrightarrow{\overrightarrow{x}} \mathcal{A}$, $\mathcal{M}, S \vDash_C \psi$ if and only if $\mathcal{A} \vDash_{PA} \theta(\overrightarrow{x}, \psi)$.*

Proof. The proof is by structural induction on ψ. Only the cases for the magic wand are detailed, the other ones being either similar or straightforward.

For the left-to-right direction, suppose $\mathcal{A} \nvDash_{\mathsf{PA}} \theta(\overrightarrow{x}, \varphi \mathrel{-\!\!*} \psi)$. There is $\mathcal{A}'$ such that $\mathcal{A}' \nvDash_{\mathsf{PA}} \Psi(\overrightarrow{y}) \to \theta(\overrightarrow{y}, \varphi) \to \Psi(\overrightarrow{z}) \to \theta(\overrightarrow{z}, \psi)$ and for all $i \in 1\,..\,n$, $\mathcal{A}'(r_i) = \mathcal{A}(r_i)$, $\mathcal{A}'(s_i) = \mathcal{A}(s_i)$, $\mathcal{A}'(x_i) = \mathcal{A}(x_i)$ and $\mathcal{A}'(z_i) = \mathcal{A}'(x_i) + \mathcal{A}'(y_i)$. Since $\mathcal{A}' \vDash_{\mathsf{PA}} \Psi(\overrightarrow{z})$, for all $i \in 1\,..\,n$, there are at least $\mathcal{A}'(y_i)$ elements $x \in X$ such that $x \in V(p_i) \setminus S$. Since $\mathcal{M}$ is atomic, there is $S_1 \subseteq X \setminus S$ such that $\mathcal{M}, S_1 \xrightarrow{\overrightarrow{y}} \mathcal{A}'$. Moreover, it can easily be checked that $\mathcal{M}, S_2 \xrightarrow{\overrightarrow{z}} \mathcal{A}'$ for $S_2 \doteq S \uplus S_1$. By induction hypothesis, $\mathcal{M}, S_1 \vDash_C \varphi$ and $\mathcal{M}, S_2 \nvDash_C \psi$. Therefore $\mathcal{M}, S \nvDash_C \varphi \mathrel{-\!\!*} \psi$.

For the right-to-left direction, suppose $\mathcal{M}, S \nvDash_C \varphi \mathrel{-\!\!*} \psi$. There are S_1, S_2 such that $S_2 = S \uplus S_1$, $\mathcal{M}, S_1 \vDash_C \varphi$ and $\mathcal{M}, S_2 \nvDash_C \psi$. Define the Presburger arithmetic assignment $\mathcal{A}'$ such that $\mathcal{A}'(r_i) = \mathcal{A}(r_i)$, $\mathcal{A}'(s_i) = \mathcal{A}(s_i)$, $\mathcal{A}'(x_i) = \mathcal{A}(x_i)$, $\mathcal{A}'(y_i) = |V(p_i) \cap S_1|$ and

$\mathcal{A}'(z_i) = |V(p_i) \cap S_2|$, for all $i \in 1..n$. By definition, $\mathcal{M}, S_1 \xrightarrow{\overrightarrow{y}} \mathcal{A}'$, $\mathcal{M}, S_2 \xrightarrow{\overrightarrow{z}} \mathcal{A}'$ and $\mathcal{A}' \vDash_{\mathsf{PA}} \bigwedge_{i \in 1..n} z_i = x_i + y_i$. By Lemma 17, $\mathcal{A}' \vDash_{\mathsf{PA}} \Psi(\overrightarrow{y}) \wedge \Psi(\overrightarrow{z})$. By induction hypothesis, $\mathcal{A}' \vDash_{\mathsf{PA}} \theta(\overrightarrow{y}, \varphi)$ and $\mathcal{A}' \nvDash_{\mathsf{PA}} \theta(\overrightarrow{z}, \psi)$. Therefore, $\mathcal{A} \nvDash_{\mathsf{PA}} \theta(\overrightarrow{x}, \varphi \twoheadrightarrow \psi)$. ◀

We can now prove the complexity upper bound for $\mathsf{CL}^{\twoheadrightarrow}_{\mathrm{atom}}$.

▶ **Proposition 19.** *The satisfiability problem of $\mathsf{CL}^{\twoheadrightarrow}_{atom}$ is in 3EXPTIME.*

Proof. Given a formula $\varphi_0 \in \Phi^{\twoheadrightarrow}$, let us choose $\overrightarrow{p}$, $\overrightarrow{r}$ and $\overrightarrow{s}$ as specified previously and $\overrightarrow{x}$ such that $\overrightarrow{x}$, $\overrightarrow{r}$ and $\overrightarrow{s}$ are pairwise disjoint. The constraint $\Psi(\overrightarrow{x}) \wedge \theta(\overrightarrow{x}, \varphi_0)$ can be computed in time polynomial in $|\varphi_0|$. If φ_0 is $\mathsf{CL}^{\twoheadrightarrow}_{\mathrm{atom}}$-satisfiable, by Lemma 16, there is a model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}_{\mathrm{atom}}$, a subset $S \in \mathcal{O}$ and a Presburger arithmetic assignment $\mathcal{A}$ such that $\mathcal{M}, S \vDash_C \varphi_0$ and $\mathcal{M}, S \xrightarrow{\overrightarrow{x}} \mathcal{A}$. By Lemmas 17 and 18, $\mathcal{A} \vDash_{\mathsf{PA}} \Psi(\overrightarrow{x}) \wedge \theta(\overrightarrow{x}, \varphi_0)$. Conversely, suppose that $\mathcal{A} \vDash_{\mathsf{PA}} \Psi(\overrightarrow{x}) \wedge \theta(\overrightarrow{x}, \varphi_0)$. By Lemma 17, there is a model $\mathcal{M} = (X, \mathcal{O}, V)$ in $\mathcal{C}_{\mathrm{atom}}$ and a subset $S \in \mathcal{O}$ such that $\mathcal{M}, S \xrightarrow{\overrightarrow{x}} \mathcal{A}$. By Lemma 18, $\mathcal{M}, S \vDash_C \varphi_0$. Therefore, there is a polynomial reduction from the satisfiability problem of $\mathsf{CL}^{\twoheadrightarrow}_{\mathrm{atom}}$ to the satisfiability problem in Presburger arithmetic. Since the satisfiability problem in Presburger arithmetic has been proved in [11] to be decidable in 3EXPTIME, the satisfiability of $\mathsf{CL}^{\twoheadrightarrow}_{\mathrm{atom}}$ is in 3EXPTIME too. ◀

7.2 Complexity of the satisfiability problem of $\mathsf{CL}^{\twoheadrightarrow}_{\mathsf{comp}}$

We provide an exponential reduction of the satisfiability problem of $\mathsf{CL}^{\twoheadrightarrow}_{\mathrm{comp}}$ to the satisfiability problem of $\mathsf{CL}^{\twoheadrightarrow}_{\mathrm{atom}}$. We first define a *power correspondence* as a triple (Q_1, b, Q_2) such that Q_1 and Q_2 are subsets of Φ_0 and b is a bijection assigning an element of Q_2 to every subset of Q_1. Then, given a power correspondence (Q_1, b, Q_2) and a formula $\varphi \in \Phi^{\twoheadrightarrow}$ such that the set of propositional variables occurring in φ is included in Q_1, the translation $\tau\,(Q_1, b, Q_2)(\varphi)$ of φ is obtained by replacing each occurrence of p in φ with

$$\bigvee_{p \in P \subseteq Q_1} b(P)$$

for all $p \in Q_1$. The resulting formula has size exponential in the size of the original formula. Moreover, given any universe X and any power correspondence (Q_1, b, Q_2), the relation $\frac{Q_1, b, Q_2}{X}$ over valuations on X is defined such that $V_1 \frac{Q_1, b, Q_2}{X} V_2$ iff for all $x \in X$ and all $Q \subseteq Q_1$, $x \in V_2\,(b\,(Q))$ iff $Q = \{p \in Q_1 \mid x \in V_1(p)\}$. We first state the following lemmas.

▶ **Lemma 20.** *For any subset model $\mathcal{M} = (X, \mathcal{O}, V_1)$ in $\mathcal{C}_{comp}$ and any power correspondence (Q_1, b, Q_2), there is a valuation V_2 on X such that $V_1 \frac{Q_1, b, Q_2}{X} V_2$ and $(X, \mathcal{O}, V_2)$ is a subset model in $\mathcal{C}_{atom}$.*

Proof. Define V_2 such that, for all $p \in \Phi_0$, $V_2(p) = \left\{x \in X \;\middle|\; b^{-1}(p) = V_1^{-1}(x)\right\}$ if $p \in Q_2$ and $V_2(p) = \emptyset$ otherwise. It can easily be checked that $V_1 \frac{Q_1, b, Q_2}{X} V_2$ and $(X, \mathcal{O}, V_2)$ is a subset model in $\mathcal{C}_{\mathrm{atom}}$. ◀

▶ **Lemma 21.** *For all subset model $\mathcal{M} = (X, \mathcal{O}, V_2)$ in $\mathcal{C}_{atom}$ and all power correspondence (Q_1, b, Q_2), there is a valuation V_1 on X such that $V_1 \frac{Q_1, b, Q_2}{X} V_2$ and $(X, \mathcal{O}, V_1)$ is a subset model in $\mathcal{C}_{comp}$.*

Proof. Define V_1 by

$$V_1(p) \doteq \bigcup_{p \in P \subseteq Q_1} V_2\,(b\,(P))\,, \text{ for all } p \in \Phi_0.$$

It can easily be checked that $V_1 \frac{Q_1, b, Q_2}{X} V_2$ and $(X, \mathcal{O}, V_1)$ is a subset model in $\mathcal{C}_{\mathrm{comp}}$. ◀

▶ **Lemma 22.** *For any subset models $\mathcal{M}_1 = (X, \mathcal{O}, V_1)$ and $\mathcal{M}_2 = (X, \mathcal{O}, V_2)$, any power correspondence (Q_1, b, Q_2) such that $V_1 \frac{Q_1,b,Q_2}{X} V_2$, any subset $S \in \mathcal{O}$ and any formula $\varphi \in \Phi^{\multimap}$ such that the set of propositional variables occurring in φ is included in Q_1, $\mathcal{M}_1, S \vDash \varphi$ if and only if $\mathcal{M}_2, S \vDash \tau(Q_1, b, Q_2)(\varphi)$.*

Proof. The proof is by a straightforward induction on $|\varphi|$, left to the reader. ◀

▶ **Proposition 23.** *The satisfiability problem of $\mathsf{CL}^{\multimap}_{comp}$ is in 4EXPTIME.*

Proof. Suppose now that we want to check whether a formula $\varphi_0 \in \Phi^{\multimap}$ is satisfiable in $\mathsf{CL}^{\multimap}_{\mathrm{comp}}$. Since Φ_0 is infinite, we can construct a power correspondence (Q_1, b, Q_2) such that Q_1 is the set of all the propositional variables occurring in φ_0. Then we check whether $\tau(Q_1, b, Q_2)(\varphi)$ is satisfiable in $\mathsf{CL}^{\multimap}_{\mathrm{atom}}$. If it is the case, then there is a model $\mathcal{M}_2$ in $\mathcal{C}_{\mathrm{atom}}$ satisfying $\tau(Q_1, b, Q_2)(\varphi)$. By Lemmas 21 and 22, there is a model $\mathcal{M}_1$ in $\mathcal{C}_{\mathrm{atom}}$ satisfying φ. Therefore, our procedure is complete. Conversely, if there is a model $\mathcal{M}_1$ in $\mathcal{C}_{\mathrm{comp}}$ satisfying φ then, by Lemmas 20 and 22, there is a model $\mathcal{M}_2$ in $\mathcal{C}_{\mathrm{comp}}$ satisfying $\tau(Q_1, b, Q_2)(\varphi)$. Therefore, our procedure is sound. Finally, computing (Q_1, b, Q_2) and $\tau(Q_1, b, Q_2)(\varphi)$ takes deterministic exponential time and the satisfiability problem of $\mathsf{CL}^{\multimap}_{\mathrm{atom}}$ can be decided in triple exponential time in the size of the input formula. Since the size of $\tau(Q_1, b, Q_2)(\varphi)$ is exponential in the size of φ, the satisfiability problem of $\mathsf{CL}^{\multimap}_{\mathrm{comp}}$ is in 4EXPTIME. ◀

▶ **Corollary 24.** *The satisfiability problem of bCL is in 4EXPTIME.*

Proof. We prove that bCL is the magic wand-free fragment of $\mathsf{CL}^{\multimap}_{\mathrm{comp}}$. Clearly, any model in $\mathcal{C}_{\mathrm{comp}}$ is in $\mathcal{C}_{\mathrm{all}}$. Suppose that φ is bCL-satisfiable. By Prop. 8, there is a finite model $\mathcal{M} = (X, \mathcal{O}, V)$ such that $\mathcal{M}, S \vDash_C \varphi$ for some $S \in \mathcal{O}$. It can easily be checked that $\mathcal{M}' \doteq (S, \mathcal{P}(S), V\restriction_S)$ is in $\mathcal{C}_{\mathrm{comp}}$ and that $\mathcal{M}', S \vDash \varphi$. ◀

8 Conclusion

We have proposed a new family of logics with an associative binary modality, called counting logics. We have shown that these logics can be seen both as specializations of the Boolean logic of bunched implications and as generalizations of the propositional dependence logic. We have also proved that the validity problem of the basic counting logic bCL is decidable, NEXPTIME-hard and in 4EXPTIME. We conjecture that this decidability result is due to particular constraints on the valuation of propositional variable and we have proved that the logic obtained by removing these constraints is undecidable. But the present work is exploratory and many problems remain open. First, the exact complexity of the basic counting logic is still unknown. Another avenue for future work is to further explore the connection between counting logic and propositional team logics, in particular logics that lack the locality property like the propositional independence logic. Finally, the most challenging problem we have left open is the decidability status of $\mathsf{bCL}^{\multimap}$, the basic counting logic with magic wands. We believe that this problem is difficult and is strongly related to the subset finite problem which consists in determining whether there is a $\mathsf{bCL}^{\multimap}$ formula that can only be satisfied at infinite subsets.

Acknowledgements. The author would like to thanks Tinko Tinchev for the numerous discussions which greatly helped us to obtain the results presented in the present article.

References

1. Samson Abramsky and Jouko A. Väänänen. From IF to BI. *Synthese*, 167(2):207–230, 2009. doi:10.1007/s11229-008-9415-6.
2. Robert Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66, 1966. doi:10.1090/memo/0066.
3. James Brotherston and Max I. Kanovich. Undecidability of propositional separation logic and its neighbours. In *Logic in Computer Science – LICS*, pages 130–139. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.24.
4. Cristiano Calcagno, Hongseok Yang, and Peter W. O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In *Foundations of Software Technology and Theoretical Computer Science – FSTTCS*, volume 2245 of *LNCS*, pages 108–119. Springer, 2001. doi:10.1007/3-540-45294-X_10.
5. Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Symposium on Principles of Programming Languages – POPL*, pages 365–377. ACM, 2000. doi:10.1145/325694.325742.
6. David Harel. Recurring dominoes: Making the highly undecidable highly understandable (preliminary report). In *Fundamentals of Computation Theory – FCT*, volume 158 of *LNCS*, pages 177–194, 1983. doi:10.1007/3-540-12689-9_103.
7. Wilfrid Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, 5(4):539–563, 1997. doi:10.1093/jigpal/5.4.539.
8. Ágnes Kurucz, István Németi, Ildikó Sain, and András Simon. Decidable and undecidable logics with a binary modality. *Journal of Logic, Language and Information*, 4(3):191–206, 1995. doi:10.1007/BF01049412.
9. Dominique Larchey-Wendling and Didier Galmiche. The undecidability of boolean BI through phase semantics. In *Logic in Computer Science – LICS*, pages 140–149. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.18.
10. Leopold Löwenheim. Über möglichkeiten im relativkalkül. *Mathematische Annalen*, 76(4):447–470, 1915. translated in [15].
11. Derek C. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978. doi:10.1016/0022-0000(78)90021-1.
12. Paritosh K. Pandya. Some extensions to propositional mean-value caculus: Expressiveness and decidability. In *Computer Science Logic – CSL*, volume 1092 of *LNCS*, pages 434–451. Springer, 1995. doi:10.1007/3-540-61377-3_52.
13. David J Pym. *The semantics and proof theory of the logic of bunched implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002.
14. John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Logic in Computer Science – LICS*, pages 55–74. IEEE Computer Society, 2002. doi:10.1109/LICS.2002.1029817.
15. Jean van Heijenoort. *From Frege to Gödel: a source book in mathematical logic, 1879-1931*, volume 9. Harvard University Press, 1967.
16. Yde Venema. A modal logic for chopping intervals. *Journal of Logic and Computation*, 1(4):453–476, 1991. doi:10.1093/logcom/1.4.453.
17. Jonni Virtema. Complexity of validity for propositional dependence logics. In *Games, Automata, Logics and Formal Verification – GandALF*, volume 161 of *EPTCS*, pages 18–31, 2014. doi:10.4204/EPTCS.161.5.
18. Fan Yang and Jouko Väänänen. Propositional logics of dependence. *Annals of Pure and Applied Logic*, 167(7):557–589, 2016. doi:10.1016/j.apal.2016.03.003.
19. Fan Yang and Jouko Väänänen. Propositional team logics. *Annals of Pure and Applied Logic*, 2017. doi:10.1016/j.apal.2017.01.007.

Noetherian Quasi-Polish Spaces*

Matthew de Brecht[1] and Arno Pauly[2]

1 **Graduate School of Human and Environmental Studies, Kyoto University, Kyoto, Japan**
matthew@i.h.kyoto-u.ac.jp
2 **Départment d'Informatique, Université libre de Bruxelles, Brussels, Belgium**
arno.m.pauly@gmail.com

Abstract

In the presence of suitable power spaces, compactness of $\mathbf{X}$ can be characterized as the singleton $\{X\}$ being open in the space $\mathcal{O}(\mathbf{X})$ of open subsets of $\mathbf{X}$. Equivalently, this means that universal quantification over a compact space preserves open predicates.

Using the language of represented spaces, one can make sense of notions such as a $\mathbf{\Sigma}^0_2$-subset of the space of $\mathbf{\Sigma}^0_2$-subsets of a given space. This suggests higher-order analogues to compactness: We can, e.g. , investigate the spaces $\mathbf{X}$ where $\{X\}$ is a $\mathbf{\Delta}^0_2$-subset of the space of $\mathbf{\Delta}^0_2$-subsets of $\mathbf{X}$. Call this notion ∇-compactness. As $\mathbf{\Delta}^0_2$ is self-dual, we find that both universal and existential quantifier over ∇-compact spaces preserve $\mathbf{\Delta}^0_2$ predicates.

Recall that a space is called Noetherian iff every subset is compact. Within the setting of Quasi-Polish spaces, we can fully characterize the ∇-compact spaces: A Quasi-Polish space is Noetherian iff it is ∇-compact. Note that the restriction to Quasi-Polish spaces is sufficiently general to include plenty of examples.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Descriptive set theory, synthetic topology, well-quasi orders, Noetherian spaces, compactness

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.16

1 Introduction

1.1 Noetherian spaces

▶ **Definition 1.** A topological space $\mathbf{X}$ is called *Noetherian*, iff every strictly ascending chain of open sets is finite.

Noetherian spaces were first studied in algebraic geometry. Here, the prime motivation is that the Zariski topology on the spectrum of a Noetherian commutative ring is Noetherian (which earns the Noetherian spaces their name).

The relevance of Noetherian spaces for computer science was noted by Goubault-Larrecq [14], based on their relationship to well quasiorders. Via well-structured transition systems [13], well quasiorders are used in verification to prove decidability of termination and related properties. Unfortunately, well quasiorders lack some desirable closure properties (the standard counterexample is due to Rado [33]), which led to the introduction of better quasiorders by Nash-Williams [25], which is a more restrictive notion avoiding the shortcomings of well quasiorders.

* This work was supported by JSPS Core-to-Core Program, A. Advanced Research Networks. The first author was supported by JSPS KAKENHI Grant Number 15K15940. The second author was supported by the ERC inVEST (279499) project.

Noetherian spaces generalize well-quasi orders: The Alexandrov topology on a quasi-order is Noetherian iff the quasi-order is a well-quasi order. As shown by Goubault-Larrecq [17], results on the preservation of well-quasi orders under various constructions (such as Higman's Lemma or Kruskal's Tree Theorem [16]) extend to Noetherian spaces; furthermore, Noetherian spaces exhibit some additional closure properties, e.g. the Hoare space of a Noetherian space is Noetherian again [14]. The usefulness of Noetherian spaces for verification is detailed by Goubault-Larrecq in [15].

1.2 Quasi-Polish spaces

A countably-based topological space is called quasi-Polish if its topology can be derived from a Smyth-complete quasi-metric. Quasi-Polish spaces were introduced by de Brecht in [6] as a joint generalization of Polish spaces and ω-continuous domains in order to satisfy the desire for a unified setting for descriptive set theory in those areas (expressed e.g. by Selivanov [36]).

1.3 Synthetic DST

Synthetic descriptive set theory as proposed by the authors in [32] reinterprets descriptive set theory in a category-theoretic context. In particular, it provides notions of lifted counterparts to topological concepts such as open sets (e.g. $\mathbf{\Sigma}$-classes from descriptive set theory), compactness, and so on.

1.4 Our contributions

In the present paper, we will study Noetherian quasi-Polish spaces. As our main result, we show that in the setting of quasi-Polish spaces, being Noetherian is the $\mathbf{\Delta}^0_2$-analogue to compactness. We present the result in two different incarnations:

Theorem 10 states the result in the language of traditional topology, i.e. a quasi-Polish space is Noetherian iff every $\mathbf{\Delta}^0_2$-cover has a finite subcover. While we will prove Theorem 10 via the Baire category theorem, it is also a simple consequence of results on the Skula topology for sober spaces.

Our second version (Theorem 39) can be seen as lifting the characterization of compactness by the projections being closed maps. To be able to even formulate this result, we employ the language of synthetic topology. This requires us to define a computable version of being Noetherian (Definition 35). A significant improvement of usefulness of Theorem 39 compared to Theorem 10 is a particular consequence: Universal and existential quantification over Noetherian spaces preserve $\mathbf{\Delta}^0_2$-predicates – and this characterizes Noetherian spaces (Proposition 43).

1.5 Structure of the article

In Section 2 we recall some results on Noetherian spaces and on quasi-Polish spaces, and then prove some observations on Noetherian quasi-Polish spaces. In particular, Theorem 10 shows that for quasi-Polish spaces, being Noetherian is equivalent to any $\mathbf{\Delta}^0_2$-cover admitting a finite subcover. This section requires only some basic background from topology.

Section 3 introduces the additional background material we need for the remainder of the paper, in particular from computable analysis and synthetic topology.

In Section 4 we investigate how Noetherian spaces ought to be defined in synthetic topology (Escardó [10]), specifically in the setting of the category of represented spaces (Pauly [29]).

Our main result will be presented in Section 5: The Noetherian spaces can be characterized amongst the quasi-Polish spaces as those allowing quantifier elimination over $\mathbf{\Delta}^0_2$-statements (Theorem 39 and Corollary 44). The core idea is that just as compact spaces are characterized by $\{X\}$ being an open subset of the space $\mathcal{O}(\mathbf{X})$ of open subsets, the Noetherian spaces are (amongst the quasi-Polish) characterized by $\{X\}$ being a $\mathbf{\Delta}^0_2$-subset of the space of $\mathbf{\Delta}^0_2$-subsets.

An extended version is available on the arXiv as [8].

2 Initial observations on Noetherian quasi-Polish spaces

2.1 Background on Quasi-Polish spaces

Recall that a quasi-metric on $\mathbf{X}$ is a function $d : \mathbf{X} \times \mathbf{X} \to [0, \infty)$ such that $x = y \Leftrightarrow d(x, y) = d(y, x) = 0$ and $d(x, z) \leq d(x, y) + d(y, z)$, i.e. a quasi-metric has all properties of a metric except symmetry. A sequence $(x_n)_{n\in\mathbb{N}}$ in a quasi-metric space is Cauchy, if $\forall k \in \mathbb{N}\ \exists N \in \mathbb{N}\ \forall m \geq n\ d(x_n, x_m) < 2^{-k}$. After Smyth [37], we call a quasi-metric space Smyth-complete, if every Cauchy sequence converges w.r.t. the metric d' obtained as $d'(x, y) := \max\{d(x, y), d(y, x)\}$.

A quasi-metric induces a topology via the basis $(B(x, 2^{-k}) := \{y \in \mathbf{X} \mid d(x, y) < 2^{-k}\})_{x\in\mathbf{X},k\in\mathbb{N}}$. A topological space is called quasi-Polish, if it is countably-based and the topology can be obtained from a Smyth-complete quasi-metric. For details we refer to [6], and only recall some select results to be used later on here.

The $\mathbf{\Sigma}^0_2$-subsets of a quasi-Polish space are those of the form $\bigcup_{n\in\mathbb{N}} U_n \setminus V_n$, where U_n and V_n are open. Complements of $\mathbf{\Sigma}^0_2$-sets are called $\mathbf{\Pi}^0_2$-subset. A set is $\mathbf{\Delta}^0_2$ iff it is both $\mathbf{\Sigma}^0_2$ and $\mathbf{\Pi}^0_2$. It follows that the $\mathbf{\Sigma}^0_2$-sets are exactly the countable unions of $\mathbf{\Delta}^0_2$-sets.

▶ **Proposition 2** (de Brecht [6]). A subspace of a quasi-Polish space is a quasi-Polish space iff it is a $\mathbf{\Pi}^0_2$-subspace.

▶ **Proposition 3** (de Brecht [6]). A space is quasi-Polish iff it is homoeomorphic to a $\mathbf{\Pi}^0_2$-subspace of the Scott domain $\mathcal{P}(\omega)$.

▶ **Theorem 4** (Heckmann [19], Becher & Grigorieff [1, Theorem 3.14]). *Let $\mathbf{X}$ be quasi-Polish. If $\mathbf{X} = \bigcup_{i\in\mathbb{N}} A_i$ with each A_i being $\mathbf{\Sigma}^0_2$, then there is some i_0 such that A_{i_0} has non-empty interior.*

Recall that a non-empty closed set is called *irreducible*, if it is not the union of two proper closed subsets. A topological space is called *sober*, if each irreducible closed set is the closure of a unique singleton.

▶ **Proposition 5** (de Brecht [6]). A countably-based locally compact sober space is quasi-Polish. Conversely, each quasi-Polish space is sober.

2.2 Background on Noetherian spaces

▶ **Theorem 6** (Folklore, cf. Goubault-Larrecq [17]). *The following are equivalent for a topological space $\mathbf{X}$:*

1. *$\mathbf{X}$ is Noetherian, i.e. every strictly ascending chain of open sets is finite (Definition 1).*
2. *Every strictly descending chain of closed sets is finite.*
3. *Every open set is compact.*
4. *Every subset is compact.*

As being Noetherian is preserved by sobrification[1], we do not lose much by restricting our attention to sober Noetherian spaces.

▶ **Lemma 7** (Folklore, cf. Goubault-Larrecq [17]). *Every closed subset of a sober Noetherian space is the closure of a finite set.*

2.3 Some new observations

▶ **Theorem 8.** *The following are equivalent for a sober Noetherian space* $\mathbf{X}$*:*
1. $\mathbf{X}$ *is countable.*
2. $\mathbf{X}$ *is countably-based.*
3. $\mathbf{X}$ *is quasi-Polish.*

▶ **Corollary 9.** *A subspace of a quasi-Polish Noetherian space is sober iff it is a* $\mathbf{\Pi}^0_2$*-subspace.*

Proof. Combine Theorem 8 with Proposition 2. ◀

The following theorem already showcases the link between being Noetherian and a $\mathbf{\Delta}^0_2$-analogue to compactness. Its proof is split into Lemmata 11,12 and Observation 13. The result can alternatively be obtained as corollary of a result by Hoffmann [20] on the relationship between Noetherian spaces and the Skula topology[2].

▶ **Theorem 10.** *The following are equivalent for a quasi-Polish space* $\mathbf{X}$*:*
1. $\mathbf{X}$ *is Noetherian.*
2. *Every* $\mathbf{\Delta}^0_2$*-cover of* $\mathbf{X}$ *has a finite subcover.*
3. *Every* $\mathbf{\Sigma}^0_2$*-cover of* $\mathbf{X}$ *has a finite subcover.*

▶ **Lemma 11.** *If a topological space* $\mathbf{X}$ *is not Noetherian, then it admits a countably-infinite* $\mathbf{\Delta}^0_2$*-partition.*

Proof. If $\mathbf{X}$ is not Noetherian, then there must be an infinite strictly ascending chain $(U_i)_{i\in\mathbb{N}}$ of open sets. Then $\{U_{i+1} \setminus U_i \mid i \in \mathbb{N}\} \cup \{U_0, \left(\bigcup_{i\in\mathbb{N}} U_i\right)^C\}$ constitutes a $\mathbf{\Delta}^0_2$-partition with countably-infinitely many non-trivial pieces. ◀

▶ **Lemma 12.** *Any* $\mathbf{\Delta}^0_2$*-cover of a Noetherian quasi-Polish space has a finite subcover.*

Proof. Since $\mathbf{X}$ is countable we can assume the covering is countable. By the Baire category theorem for quasi-Polish spaces (Theorem 4), there is a $\mathbf{\Delta}^0_2$-set A_0 in the covering such that its interior, U_0 is non-empty.

For $n \geq 0$, if $\mathbf{X} \neq U_n$, then we repeat the same argument with respect to $\mathbf{X} \setminus U_n$ to get a $\mathbf{\Delta}^0_2$-set A_{n+1} in the covering with non-empty interior relative to $\mathbf{X} \setminus U_n$. Define U_{n+1} to be the union of U_n and the relative interior of A_{n+1}. Then U_{n+1} is an open subset of $\mathbf{X}$ which strictly contains U_n. Since $\mathbf{X}$ is Noetherian, eventually $\mathbf{X} = U_n$, and $A_0, \ldots, A_n$ will yield a finite subcovering of $\mathbf{X}$. ◀

▶ **Observation 13.** Any $\mathbf{\Sigma}^0_2$-cover of a quasi-Polish space can be refined into a $\mathbf{\Delta}^0_2$-cover, and any $\mathbf{\Delta}^0_2$-cover is a $\mathbf{\Sigma}^0_2$-cover.

Proof. In a quasi-Polish space, every $\mathbf{\Sigma}^0_2$-set is a (countable) union of $\mathbf{\Delta}^0_2$-sets. ◀

---

[1] Sobrification only adds points, not open sets, and being Noetherian is only about open sets.
[2] Thanks to an anonymous referee for pointing this out to us. See also [17, Exercise 9.7.16.].

▶ **Corollary 14.** *Let* $\mathbf{X}$ *be a Noetherian quasi-Polish space, and let* $\mathbf{X}^\delta$ *be the topology induced by the* $\mathbf{\Delta}^0_2$*-subsets of* $\mathbf{X}$*. Then* $\mathbf{X}^\delta$ *is a compact Hausdorff space.*

Proof. That $\mathbf{X}^\delta$ is compact follows from Lemma 12. To see that it is Hausdorff, we just note that in any T_0-space, two distinct points can be separated by a disjoint pair of an open and a closed set – hence by $\mathbf{\Delta}^0_2$-sets. ◀

We can obtain the following special case of Goubault-Larrecq's Lemma 7 as a corollary of Lemma 12:

▶ **Corollary 15.** *Every closed subset of a quasi-Polish Noetherian space is the closure of a finite set.*

Proof. Given some closed subset $A \subseteq \mathbf{X}$, consider the $\mathbf{\Delta}^0_2$-cover $\mathbf{X} = A^C \cup \bigcup_{x\in A} \mathrm{cl}\{x\}$. By Lemma 12 there is some finite subcover $\mathbf{X} = A^C \cup \bigcup_{x\in F} \mathrm{cl}\{x\}$, but then it follows that $A = \mathrm{cl}F$. ◀

Neither being sober nor being quasi-Polish is preserved by continuous images in general. However, being Noetherian is not only preserved itself, but in its presence, so are the other properties:

▶ **Proposition 16.** Let $\mathbf{X}$ be a Noetherian sober (quasi-Polish) space and $\sigma : \mathbf{X} \to \mathbf{Y}$ a continuous surjection. Then $\mathbf{Y}$ is Noetherian sober (quasi-Polish) space, too.

Proof. Let $C \subseteq \mathbf{Y}$ be irreducible closed. Then $\sigma^{-1}(C)$ is closed, so by Lemma 7 (or Corollary 15) there is finite $F \subseteq \mathbf{X}$ such that $\mathrm{cl}(F) = \sigma^{-1}(C)$. Continuity implies $\mathrm{cl}(\sigma(F)) \supseteq \sigma(\mathrm{cl}(F)) = C$, hence $\mathrm{cl}(\sigma(F)) = C$. Since $\sigma(F)$ is finite and C is irreducible, C must be equal to the closure of some element of $\sigma(F)$. Therefore, $\mathbf{Y}$ is sober.

By Theorem 8, for Noetherian sober spaces being quasi-Polish is equivalent to being countable, which is clearly preserved by (continuous) surjections. ◀

3 Background

3.1 Computable analysis

In the remainder of this article, we wish to explore the uniform or effective aspects of the theory of Noetherian Quasi-Polish spaces. The basic framework for this is provided by *computable analysis* [40]. Here the core idea is to introduce notions of continuity and in particular continuity on a wide range of spaces by translating them from those on Baire space via the so-called representations. Our notation and presentation follows closely that of [29], which in turn is heavily influenced by Escardó's *synthetic topology* [10], and by work by Schröder [34].

▶ **Definition 17.** A represented space is a pair $\mathbf{X} = (X, \delta_\mathbf{X})$ where X is a set and $\delta_\mathbf{X} :\subseteq \mathbb{N}^\mathbb{N} \to X$ is a partial surjection. A function between represented spaces is a function between the underlying sets.

▶ **Definition 18.** For $f :\subseteq \mathbf{X} \to \mathbf{Y}$ and $F :\subseteq \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$, we call F a realizer of f (notation $F \vdash f$), iff $\delta_Y(F(p)) = f(\delta_X(p))$ for all $p \in \mathrm{dom}(f\delta_X)$. A map between represented spaces is called computable (continuous), iff it has a computable (continuous) realizer.

Represented spaces $\mathbf{X}$, $\mathbf{X}'$ are *computably isomorphic*, written $\mathbf{X} \cong \mathbf{X}'$, if there is a bijection $f : \mathbf{X} \to \mathbf{X}'$ such that both f and f' are computable.

Two represented spaces of particular importance are the integers $\mathbb{N}$ and Sierpiński space $\mathbb{S}$. The represented space $\mathbb{N}$ has as underlying set $\mathbb{N}$ and the representation $\delta_{\mathbb{N}} : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ defined by $\delta_{\mathbb{N}}(p) = p(0)$. The Sierpiński space $\mathbb{S}$ has the underlying set $\{\top, \bot\}$ and the representation $\delta_{\mathbb{S}}$ with $\delta_{\mathbb{S}}(0^\omega) = \bot$ and $\delta_{\mathbb{S}}(p) = \top$ for $p \neq 0^\omega$.

Represented spaces have binary products, defined in the obvious way: The underlying set of $\mathbf{X} \times \mathbf{Y}$ is $X \times Y$, with the representation $\delta_{\mathbf{X}\times\mathbf{Y}}(\langle p, q\rangle) = (\delta_{\mathbf{X}}(p), \delta_{\mathbf{Y}}(q))$. Here $\langle\ ,\ \rangle : \mathbb{N}^{\mathbb{N}} \times \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ is the pairing function defined via $\langle p, q\rangle(2n) = p(n)$ and $\langle p, q\rangle(2n+1) = q(n)$.

A central reason why the category of represented spaces is such a convenient setting lies in the fact that it is cartesian closed: We have available a function space construction $\mathcal{C}(\cdot, \cdot)$, where the represented space $\mathcal{C}(\mathbf{X}, \mathbf{Y})$ has as underlying set the continuous functions from $\mathbf{X}$ to $\mathbf{Y}$, represented in such a way that the evaluation map $(f, x) : \mathcal{C}(\mathbf{X}, \mathbf{Y}) \times \mathbf{X} \to \mathbf{Y}$ becomes computable. This can be achieved, e.g., by letting nq represent f, if the n-th Turing machine equipped with oracle q computes a realizer of f. This also makes currying, uncurrying and composition all computable maps.

Having available to us the space $\mathbb{S}$ and the function space construction, we can introduce the spaces $\mathcal{O}(\mathbf{X})$ and $\mathcal{A}(\mathbf{X})$ of open and closed subsets respectively of a given represented space $\mathbf{X}$. For this, we identify an open subset U of $\mathbf{X}$ with its (continuous) characteristic function $\chi_U : \mathbf{X} \to \mathbb{S}$, and a closed subset with the characteristic function of the complement. As countable join (or) and binary meet (and) on $\mathbb{S}$ are computable, we can conclude that open sets are uniformly closed under countable unions, binary intersections and preimages under continuous functions by merely using elementary arguments about function spaces. The space $\mathcal{A}(\mathbf{X})$ corresponds to the upper Fell topology [12] on the hyperspace of closed sets.

Note that neither negation $\neg : \mathbb{S} \to \mathbb{S}$ (i.e. mapping $\top$ to $\bot$ and $\bot$ to $\top$) nor countable meet (and) $\bigwedge : \mathcal{C}(\mathbb{N}, \mathbb{S}) \to \mathbb{S}$ (i.e. mapping the constant sequence $(\top)_{n\in\mathbb{N}}$ to $\top$ and every other sequence to $\bot$) are continuous or computable operations. They will play the role of fundamental counterexamples in the following. Both operations are equivalent to the *limited principle of omniscience* (LPO) in the sense of Weihrauch reducibility [39].

We need two further hyperspaces, which both will be introduced as subspaces of $\mathcal{O}(\mathcal{O}(\mathbf{X}))$. The space $\mathcal{K}(\mathbf{X})$ of saturated compact sets identifies $A \subseteq \mathbf{X}$ with $\{U \in \mathcal{O}(\mathbf{X}) \mid A \subseteq U\} \in \mathcal{O}(\mathcal{O}(\mathbf{X}))$. Recall that a set is saturated, iff it is equal to the intersection of all open sets containing it (this makes the identification work). The saturation of A is denoted by $\uparrow A := \bigcap\{U \in \mathcal{O}(\mathbf{X}) \mid A \subseteq U\}$. Compactness of A corresponds to $\{U \in \mathcal{O}(\mathbf{X}) \mid A \subseteq U\}$ being open itself. The dual notion to compactness is *overtness*[3]. We obtain the space $\mathcal{V}(\mathbf{X})$ of overt sets by identifying a closed set A with $\{U \in \mathcal{O}(\mathbf{X}) \mid A \cap U \neq \emptyset\} \in \mathcal{O}(\mathcal{O}(\mathbf{X}))$. The space $\mathcal{V}(\mathbf{X})$ corresponds to the lower Fell (equivalently, the lower Vietoris) topology.

Aligned with the definition of the compact and overt subsets of a space, we can also define when a space itself is compact (respectively overt):

▶ **Definition 19.** A represented space $\mathbf{X}$ is (computably) compact, iff isFull : $\mathcal{O}(\mathbf{X}) \to \mathbb{S}$ mapping X to $\top$ and any other open set to $\bot$ is continuous (computable). Dually, it is (computably) overt, iff isNonEmpty : $\mathcal{O}(\mathbf{X}) \to \mathbb{S}$ mapping $\emptyset$ to $\bot$ and any non-empty open set to $\top$ is continuous (computable).

[3] This notion is much less known than compactness, as it is classically trivial. It is crucial in a uniform perspective, though. The term *overt* was coined by Taylor [38], based on the observation that these sets share several closure properties with the open sets.

The relevance of $\mathcal{K}(\mathbf{X})$ and $\mathcal{V}(\mathbf{X})$ is found in particular in the following characterizations, which show that compactness just makes universal quantification preserve open predicates, and dually, overtness makes existential quantification preserve open predicates. We shall see later that being Noetherian has the same role for $\mathbf{\Delta}^0_2$-predicates.

▶ **Proposition 20** ([29, Proposition 40]). The map $\exists : \mathcal{O}(\mathbf{X} \times \mathbf{Y}) \times \mathcal{V}(\mathbf{X}) \to \mathcal{O}(\mathbf{Y})$ defined by $\exists(R, A) = \{y \in Y \mid \exists x \in A\ (x, y) \in R\}$ is computable. Moreover, whenever $\exists : \mathcal{O}(\mathbf{X} \times \mathbf{Y}) \times \mathcal{S}(\mathbf{X}) \to \mathcal{O}(\mathbf{Y})$ is computable for some hyperspace $\mathcal{S}(\mathbf{X})$ and some space $\mathbf{Y}$ containing a computable element y_0, then $\overline{\ } : \mathcal{S}(\mathbf{X}) \to \mathcal{V}(\mathbf{X})$ is computable, where $\overline{\ }$ denotes topological closure.

▶ **Proposition 21** ([29, Proposition 42]). The map $\forall : \mathcal{O}(\mathbf{X} \times \mathbf{Y}) \times \mathcal{K}(\mathbf{X}) \to \mathcal{O}(\mathbf{Y})$ defined by $\forall(R, A) = \{y \in Y \mid \forall x \in A\ (x, y) \in R\}$ is computable. Moreover, whenever $\forall : \mathcal{O}(\mathbf{X} \times \mathbf{Y}) \times \mathcal{S}(\mathbf{X}) \to \mathcal{O}(\mathbf{Y})$ is computable for some hyperspace $\mathcal{S}(\mathbf{X})$ and some space $\mathbf{Y}$ containing a computable element y_0, then $\uparrow \mathrm{id} : \mathcal{S}(\mathbf{X}) \to \mathcal{K}(\mathbf{X})$ is computable.

3.2 Connecting computable analysis and topology

Calling the elements of $\mathcal{O}(\mathbf{X})$ the *open sets* is justified by noting that they indeed form a topology, namely the final topology X inherits from the subspace topology of $\mathrm{dom}(\delta_{\mathbf{X}})$ along $\delta_{\mathbf{X}}$. The notion of a continuous map between the represented spaces $\mathbf{X}$, $\mathbf{Y}$ however differs from that of a continuous map between the induced topological spaces. For a large class of spaces, the notions do coincide after all, as observed originally by Schröder [35].

▶ **Definition 22.** Call $\mathbf{X}$ admissible, if the map $x \mapsto \{U \in \mathcal{O}(\mathbf{X}) \mid x \in U\} : \mathbf{X} \to \mathcal{O}(\mathcal{O}(\mathbf{X}))$ admits a continuous partial inverse.

▶ **Theorem 23** ([29, Theorem 36]). *A represented space $\mathbf{X}$ is admissible iff any map $f : \mathbf{Y} \to \mathbf{X}$ is continuous as a map between represented spaces iff it is continuous as a map between the induced topological spaces.*

The admissible represented spaces are themselves cartesian closed (in fact, it suffices for $\mathbf{Y}$ to be admissible in order to make $\mathcal{C}(\mathbf{X}, \mathbf{Y})$ admissible). They can be seen as a joint subcategory of the sequential topological spaces and the represented spaces, and thus form the natural setting for computable topology. They have been characterized by Schröder as the QCB_0-spaces [35], the T_0 quotients of countably based spaces.

Weihrauch [40, 41] introduced the standard representation of a countably based T_0 space: Given some enumeration $(U_n)_{n\in\mathbb{N}}$ of a basis of a topological space $\mathbf{X}$, one can introduce the representation $\delta_{\mathbf{B}}$ where $\delta_{\mathbf{B}}(p) = x$ iff $\{n \in \mathbb{N} \mid \exists i\ p(i) = n + 1\} = \{n \in \mathbb{N} \mid x \in U_n\}$. This yields an admissible representation, which in turn induces the original topology on $\mathbf{X}$.

Amongst the countably based spaces, the quasi-Polish spaces are distinguished by a completeness properties. We will make use of the following characterization:

▶ **Theorem 24** (dB [6]). *A topological space $\mathbf{X}$ is quasi-Polish, iff its topology is induced by an open admissible total representation $\delta_{\mathbf{X}} : \mathbb{N}^{\mathbb{N}} \to \mathbf{X}$.*

3.3 Synthetic descriptive set theory

The central addition of synthetic descriptive set theory (as proposed by the authors in [30, 32]) is the notion of a computable endofunctor:

▶ **Definition 25.** An endofunctor d on the category of represented spaces is called *computable*, if for any represented spaces $\mathbf{X}$, $\mathbf{Y}$ the induced morphism $d : \mathcal{C}(\mathbf{X}, \mathbf{Y}) \to \mathcal{C}(d\mathbf{X}, d\mathbf{Y})$ is computable.

To keep things simple, we will restrict our attention here to endofunctors that do not change the underlying set of a represented spaces, but may only modify the representation. Such endofunctors can in particular be derived from certain maps on Baire space, called *jump operators* by de Brecht in [7]. Here, we instead adopt the terminology *transparent map* introduced in [4]. Further properties of transparent maps were studied in [27].

▶ **Definition 26.** Call $T :\subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ *transparent* iff for any computable (continuous) $g :\subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ there is a computable (continuous) $f :\subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ with $T \circ f = g \circ T$.

If the required witness f can be effectively obtained from g,, then T will induce a computable endofunctor t by setting $t\mathbf{X}$ to be $(X, \delta_{\mathbf{X}} \circ T)$, and extending to functions in the obvious way.

By applying a suitable endofunctor to Sierpiński space, we can define further classes of subsets; in particular those commonly studied in descriptive set theory. This idea and its relationship to universal sets is further explored in [18]. Basically, we introduce the space $\mathcal{O}^d(\mathbf{X})$ of d-open subsets of $\mathbf{X}$ by identifying a subset U with its continuous characteristic function $\chi_U : \mathbf{X} \to d\mathbb{S}$. If d preserves countable products, it automatically follows that the d-open subsets are effectively closed under countable unions, binary intersections and preimages under continuous maps. The complements of the d-opens are the d-closed sets, denoted by $\mathcal{A}^d(\mathbb{S})$.

We will use the endofunctors to generate lifted versions of compactness and overtness:

▶ **Definition 27.** A represented space $\mathbf{X}$ is (computably) d-compact, iff isFull : $\mathcal{O}^d(\mathbf{X}) \to d\mathbb{S}$ mapping X to $\top$ and any other open set to $\bot$ is continuous (computable). Dually, it is (computably) d-overt, iff isNonEmpty : $\mathcal{O}^d(\mathbf{X}) \to d\mathbb{S}$ mapping $\emptyset$ to $\bot$ and any non-empty open set to $\top$ is continuous (computable).

A fundamental example of a computable endofunctor linked to notions from descriptive set theory is the limit or jump endofunctor;

▶ **Definition 28.** Let $\lim :\subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ be defined via $\lim(p)(n) = \lim_{i\to\infty} p(\langle n, i\rangle)$, where $\langle\ ,\ \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a standard pairing function. Define the computable endofunctor $'$ by $(X, \delta_{\mathbf{X}})' = (X, \delta_{\mathbf{X}} \circ \lim)$ and the straight-forward lift to functions.

The map lim and its relation to the Borel hierarchy and Weihrauch reducibility was studied by Brattka in [2]. The jump of a represented spaces was studied in [42, 4]. The $'$-open sets are just the $\mathbf{\Sigma}^0_2$-sets, and the further levels of the Borel hierarchy can be obtained by iterating the endofunctor.

3.3.1 Computability with finitely many mindchanges

The most important endofunctor for our investigation of Noetherian Quasi-Polish spaces is the finite mindchange endofunctor ∇:

▶ **Definition 29** ([30])**.** Define $\Delta :\subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ via $\Delta(p)(n) = p(n + 1 + \max\{i \mid p(i) = 0\}) - 1$. Let the finite mindchange endofunctor be defined via $(X, \delta_X)^\nabla = (X, \delta_X \circ \Delta)$ and $(f : \mathbf{X} \to \mathbf{Y})^\nabla = f : \mathbf{X}^\nabla \to \mathbf{Y}^\nabla$.

We find that ∇ is a monad, and moreover, that $f : \mathbf{X} \to \mathbf{Y}^\nabla$ is computable (continuous) iff $f : \mathbf{X}^\nabla \to \mathbf{Y}^\nabla$ is. The computable maps from $\mathbf{X}$ to $\mathbf{Y}^\nabla$ can equivalently be understood as those maps from $\mathbf{X}$ to $\mathbf{Y}$ that are computable with finitely many mindchanges.

A machine model for computation with finitely many mindchanges is obtained by adding the option of resetting the output tape to the initial state. To ensure that the output is well-defined, such a reset can only be used finitely many times. Essentially, each $n+1$ in the input of Δ corresponds to a *write* n command, whereas each 0 represents the *reset*-command.

In the context of computable analysis, computation with finitely many mindchanges was studied by a number of authors [42, 9, 3, 5, 26]. For our purposes, an equivalent model based on non-deterministic computation turns out to be more useful. We say that a function from $\mathbf{X}$ to $\mathbf{Y}$ is non-deterministically computable with advice space $\mathbb{N}$, if on input p (a name for some $x \in \mathbf{X}$) the machine can guess some $n \in \mathbb{N}$ and then either continue for ω many steps and output a valid name for $f(x)$, or at some finite time reject the guess. We demand that for any p there is some $n \in \mathbb{N}$ that is not rejected. The equivalence of the two models is shown in [3].

The interpretation of ∇ in descriptive set theory is related to the $\mathbf{\Delta}^0_2$-sets. In particular, the ∇-open sets are the $\mathbf{\Delta}^0_2$-sets, the continuous functions from $\mathbf{X}$ to $\mathbf{Y}^\nabla$ are the piecewise continuous functions for Polish $\mathbf{X}$, and the lifted version of admissibility under ∇ corresponds to the Jayne-Rogers theorem (cf. [21, 23, 22]). This was explored in detail by the authors in [31].

4 ∇-computably Noetherian spaces

In this section, we want to investigate the notion of being Noetherian in the setting of synthetic topology. We will see that the naive approach fails, but then provide a well-behaved definition. That it is adequate will be substantiated by providing a computable counterpart to the relationship between Noetherian spaces and well-quasiorders. First, however, we will explore a prototypical example.

4.1 A case study on computably Noetherian spaces

Let $\mathbb{N}_<$ be the natural numbers with the topology $\mathcal{T}_< := \{L_n := \{i \in \mathbb{N} \mid i \geq n\} \mid n \in \mathbb{N}\} \cup \{\emptyset\}$. Then let $\overline{\mathbb{N}}_<$ be the result of adjoining ∞, which is contained in all non-empty open sets. In $\overline{\mathbb{N}}_<$ we find a very simple yet non-trivial example of a quasi-Polish Noetherian space.

Similarly, let $\mathbb{N}_>$ be the natural numbers with the topology $\mathcal{T}_> := \{U_n := \{i \in \mathbb{N} \mid i < n\} \mid n \in \mathbb{N}\} \cup \{\mathbb{N}\}$. By $\overline{\mathbb{N}}_>$ we denote the space resulting from adjoining an element ∞, which is only contained in one open set. In terms of representations, we can conceive of an element in $\mathbb{N}_<$ as being given as the limit of an increasing sequence, and of an element in $\mathbb{N}_>$ as the limit of a decreasing sequence.

Looking at the way how we defined $\mathcal{T}_<$, we see that we have a countable basis, and given indices of open sets, can e.g. decide subset inclusion. The indexing is fully effective, in the sense that this is a computable basis as follows:

▶ **Definition 30** ([18, Definition 9])**.** An effective countable base for $\mathbf{X}$ is a computable sequence $(U_i)_{i\in\mathbb{N}} \in \mathcal{C}(\mathbb{N}, \mathcal{O}(\mathbf{X}))$ such that the multivalued partial map Base $:\subseteq \mathbf{X} \times \mathcal{O}(\mathbf{X}) \rightrightarrows \mathbb{N}$ is computable. Here $\mathrm{dom}(\mathrm{Base}) = \{(x, U) \mid x \in U\}$ and $n \in \mathrm{Base}(x, U)$ iff $x \in U_n \subseteq U$.

Even though all open sets are basis elements, we should still distinguish computability on the open sets themselves, and computability on the indices. For example, the map $\bigcup : \mathcal{O}(\mathbf{X})^{\mathbb{N}} \to \mathcal{O}(\mathbf{X})$, i.e. the countable union of open sets, should always be a computable

operation. This, however, cannot be done on the indices. More generally, in the synthetic topology framework the space of open subsets of a given space automatically comes with its own natural topology. This topology is obtained by demanding that given a point and an open set, we can recognize (semidecide) membership. In the case of $\mathbb{N}_<$, we can establish a quite convenient characterization of its open subsets:

▶ **Proposition 31.** The map $n \mapsto \{i \in \mathbb{N} \mid i \geq n\} : \overline{\mathbb{N}}_> \to \mathcal{O}(\mathbb{N}_<)$ is a computable isomorphism.

Proof.

1. The map is computable.
   Given $m \in \mathbb{N}_<$ and $n \in \overline{\mathbb{N}}_>$, we can semidecide $m \geq n$ (just wait until the increasing and the decreasing approximations pass each other).
2. The map is surjective.
   At the moment some number m is recognized to be an element of some open set $U \in \mathcal{O}(\mathbb{N}_<)$, we have only learned some lower bound on m so far. Thus, any number greater than m is contained in U, too. Hence all open subsets of $\mathbb{N}_<$ are final segments.
3. The inverse of the map is computable.
   Given $U \in \mathcal{O}(\mathbb{N}_<)$, we can simultaneously begin testing $i \in U$? for all $i \in \mathbb{N}$. Any positive test provides an upper bound for the n such that $U = \{i \in \mathbb{N} \mid i \geq n\}$. ◀

The space of (saturated) compact subsets likewise comes with its own topology, in this case obtained by demanding that given a compact K and an open U, we can recognize if $K \subseteq U$. Similarly to the preceding proposition, we can also characterize the compact subsets of $\mathbb{N}_>$:

▶ **Proposition 32.** The map $n \mapsto \{i \in \mathbb{N} \mid i \geq n\} : \overline{\mathbb{N}}_< \to \mathcal{K}(\mathbb{N}_<)$ is a computable isomorphism.

Proof.

1. The map is computable.
   We need to show that given $n \in \overline{\mathbb{N}}_<$ and $U \in \mathcal{O}(\mathbb{N}_>)$ we can recognize that $\{i \in \mathbb{N} \mid i \geq n\} \subseteq U$. By Proposition 31, we can assume that U is of the form $U = \{i \in \mathbb{N} \mid i \geq m\}$ with $m \in \overline{\mathbb{N}}_>$. Now for such n, m, we can indeed semidecide $m \leq n$ – again, just wait until the approximating sequences reach the same value.
2. The map is surjective.
   While any subset of $\mathbb{N}_<$ is compact, only the saturated compact sets appear in $\mathcal{K}(\mathbb{N}_>)$, and these are the given ones.
3. The inverse map is computable.
   Given a compact set $K \in \mathcal{K}(\mathbb{N}_<)$, we simultaneously test if it is covered by open sets of the form $\{i \mid i \geq m\}$. Any such m we find provides a lower bound for the n for which $K = \{i \mid i \geq n\}$ holds. ◀

So we see that while the **spaces** $\mathcal{O}(\mathbb{N}_<)$ and $\mathcal{K}(\mathbb{N}_<)$ contain the same points, their topologies differ – and are, in fact, incomparable. There are two potential ways to capture the idea that *opens are compact* in a synthetic way:

We could work with open and compact sets when in a Noetherian space, i.e. with the space $\mathcal{O}(\mathbb{N}_<) \wedge \mathcal{K}(\mathbb{N}_<)$ carrying the join of the topologies. As $\mathbb{N}_< \wedge \mathbb{N}_> \cong \mathbb{N}$, in this special case we would end up in the same situation as using computability on base indices straightaway. In general though it is not even obvious if $\cap : (\mathcal{O}(\mathbf{X}) \wedge \mathcal{K}(\mathbf{X})) \times (\mathcal{O}(\mathbf{X}) \wedge \mathcal{K}(\mathbf{X})) \to (\mathcal{O}(\mathbf{X}) \wedge \mathcal{K}(\mathbf{X}))$ should be computable.

The second approach relies on the observation that $\mathbb{N}_<$ and $\mathbb{N}_>$ do not differ by *that much*. We can consider *computability with finitely many mindchanges* – and the distinction between $\mathbb{N}_<$, $\mathbb{N}_>$ and $\mathbb{N}$ disappears, as we find $\mathbb{N}_<^\nabla \cong \mathbb{N}_>^\nabla \cong \mathbb{N}^\nabla$. As the next subsection shows, computability with finitely many mindchanges seems adequate to give *opens are compact* a computable interpretation.

4.2 The abstract approach

The straightforward approach to formulate a synthetic topology version of Noetherian would be the following:

▶ **Definition 33** (Hypothetical)**.** Call a space $\mathbf{X}$ *computably Noetherian*, iff $\mathrm{id}_{\mathcal{O},\mathcal{K}} : \mathcal{O}(\mathbf{X}) \to \mathcal{K}(\mathbf{X})$ is well-defined and computable.

This fails entirely, though:

▶ **Observation 34.** Let $\mathbf{X}$ be non-empty. Then $\mathbf{X}$ is not computably Noetherian according to Definition 33.

Proof. Note that $\subseteq : \mathcal{K}(\mathbf{X}) \times \mathcal{O}(\mathbf{X}) \to \mathbb{S}$ is by definition of $\mathcal{K}$ a computable map, i.e. inclusion of a compact in an open set is semidecidable. Furthermore, $\iota : \mathbb{S} \to \mathcal{O}(\mathbf{X})$ defined via $\iota(\top) = X$ and $\iota(\bot) = \emptyset$ is a always a computable injection for non-empty $\mathbf{X}$. Now if $\mathbf{X}$ were computably Noetherian, then the map $t \mapsto \subseteq(\mathrm{id}_{\mathcal{O},\mathcal{K}}(\iota(t)), \emptyset)$ would be computable and identical to $\neg : \mathbb{S} \to \mathbb{S}$, but the latter is non-computable. ◀

We can avoid this problem by relaxing the computability-requirement to computability with finitely many mindchanges. Now we can try again:

▶ **Definition 35.** Call a space $\mathbf{X}$ ∇-*computably Noetherian*, iff $\mathrm{id}_{\mathcal{O},\mathcal{K}} : \mathcal{O}(\mathbf{X}) \to (\mathcal{K}(\mathbf{X}))^\nabla$ is well-defined and computable.

Say that an effective countable base is *nice*, if $\{\langle u, v\rangle \mid \big(U_{u(1)} \cup \ldots \cup U_{u(|u|)}\big) \subseteq \big(U_{v(1)} \cup \ldots \cup U_{v(|v|)}\big)\}$ is a decidable subset of $\mathbb{N}^* \times \mathbb{N}^*$. Clearly any effective countable base is nice relative to some oracle, hence this requirement is unproblematic from the perspective of continuity.

We can now state and prove the following theorem, which can be seen as a uniform counterpart to Theorem 6:

▶ **Theorem 36.** *Let $\mathbf{X}$ be quasi-Polish, and in particular have a nice effective countable base. Then the following are equivalent:*

1. *$\mathbf{X}$ is ∇-computably Noetherian*
2. *$\mathrm{id}_{\mathcal{O},\mathcal{K}} : \mathcal{O}(\mathbf{X}) \to (\mathcal{K}(\mathbf{X}))^\nabla$ is well-defined and computable.*
3. *$\subseteq : \mathcal{O}(\mathbf{X}) \times \mathcal{O}(\mathbf{X}) \to \mathbb{S}^\nabla$ is computable.*
4. *$\mathrm{Stabilize} : \mathcal{C}(\mathbb{N}, \mathcal{O}(\mathbf{X})) \rightrightarrows \mathbb{N}^\nabla$ is well-defined and computable, where $N \in \mathrm{Stabilize}((V_i)_{i\in\mathbb{N}})$ iff $\left(\bigcup_{i=0}^N V_i\right) = \left(\bigcup_{i\in\mathbb{N}} V_i\right)$.*
5. *$\mathrm{Stabilize} : \mathcal{C}(\mathbb{N}, \mathcal{A}(\mathbf{X})) \rightrightarrows \mathbb{N}^\nabla$ is well-defined and computable, where $N \in \mathrm{Stabilize}((A_i)_{i\in\mathbb{N}})$ iff $\left(\bigcap_{i=0}^N A_i\right) = \left(\bigcap_{i\in\mathbb{N}} A_i\right)$.*
6. *The computable map $u \mapsto \big(U_{u(1)} \cup \ldots \cup U_{u(|u|)}\big) : \mathbb{N}^* \to \mathcal{O}(\mathbf{X})$ is a surjection and has a ∇-computable right-inverse.*

Note that the forward implications hold for arbitrary represented spaces, as long as they make sense.

Proof.

1. ⇔ 2. This is the definition.

2. ⇒ 3. By taking into account the definition of $\mathcal{K}$, we have $\mathrm{id}_{\mathcal{O},\mathcal{K}} : \mathcal{O}(\mathbf{X}) \to (\mathcal{C}(\mathcal{O}(\mathbf{X}), \mathbb{S}))^{\nabla}$. Moreover, $\mathrm{id} : \mathcal{C}(\mathbf{Y}, \mathbf{Z})^{\nabla} \to \mathcal{C}(\mathbf{Y}, \mathbf{Z}^{\nabla})$ is always computable, so currying yields the claim.

3. ⇒ 4. First, we prove that Stabilize is well-defined. Assume that it is not, then there is a family $(V_i)_{i\in\mathbb{N}}$ of open sets such that $V := \bigcup_{i\in\mathbb{N}} V_i \neq \bigcup_{i=0}^{N} V_i$ for all $N \in \mathbb{N}$. Consider the computable map $q \mapsto \subseteq \left(V, \bigcup_{i\in\mathbb{N}} V_{q(i)}\right) : \mathbb{N}^{\mathbb{N}} \to \mathbb{S}^{\nabla}$. If the range of q is finite, then the output must be $\bot$, if the range of q is $\mathbb{N}$, then the output must be $\top$. However, these two cases cannot be distinguished in a $\mathbf{\Delta}^0_2$-way, thus the $(V_i)_{i\in\mathbb{N}}$ cannot exist, and Stabilize is well-defined.

To see that we can compute the (multivalued) inverse, we employ the equivalence to ∇-computability and non-deterministic computation with advice space $\mathbb{N}$ from [3]. Given $(V_i)_{i\in\mathbb{N}}$, we guess $N \in \mathbb{N}$ together with an upper bound b on the number of mindchanges happening in verifying that $\subseteq(\bigcup_{i\in\mathbb{N}} V_i, \bigcup_{i=0}^{N} V_i) = \top$. Any correct guess contains a valid solution, and any wrong guess can be rejected.

4. ⇔ 5. By de Morgan's law.

4. ⇒ 6. In a quasi-Polish space $\mathbf{X}$ with effectively countable basis $(U_i)_{i\in\mathbb{N}}$, any $U \in \mathcal{O}(\mathbf{X})$ can be effectively represented by $p \in \mathbb{N}^{\mathbb{N}}$ with $U = \bigcup_{i\in\mathbb{N}} U_{p(i)}$. Applying stabilize to the family $(U_{p(i)})_{i\in\mathbb{N}}$ shows subjectivity and computability of the multivalued inverse.

6. ⇒ 2. First we argue that (6.) implies that Stabilize from (4.) is well-defined. For the sake of a contradiction, assume that $p \in \mathbb{N}^{\mathbb{N}}$ is such that $\bigcup_{i\in\mathbb{N}} U_{p(i)} \neq \bigcup_{i\leq n} U_{p(i)}$ for all $n \in \mathbb{N}$; and that I is a ∇-computable realizer of the inverse to $u \mapsto \left(U_{u(1)} \cup \ldots \cup U_{u(|u|)}\right) : \mathbb{N}^* \to \mathcal{O}(\mathbf{X})$. We show how to construct some $q \in \mathbb{N}^{\mathbb{N}}$ such that I changes its mind infinitely often on q. We start copying p to q. At any stage, the current output of I cannot be both equal to $\bigcup_{i\in\mathbb{N}} U_{p(i)}$ and $\bigcup_{i\leq n} U_{p(i)}$ (for any $n \in \mathbb{N}$. If it is the former, we extend q by copying its last entry until I changes its mind. If I never changed its mind, then I answered wrong. If it is the latter, we copy p to q until I changes its mind. Again, if I never changed its mind, then I answered wrong.

Next we conclude that $u \mapsto \left(U_{u(1)} \cup \ldots \cup U_{u(|u|)}\right) : \mathbb{N}^* \to \mathcal{K}(\mathbf{X})$ is computable, provided that $(U_n)_{n\in\mathbb{N}}$ is a nice basis. For this, note that given $u \in \mathbb{N}^*$ and $p \in \mathbb{N}^{\mathbb{N}}$, we can semidecide whether $\left(U_{u(1)} \cup \ldots \cup U_{u(|u|)}\right) \subseteq \bigcup_{n\in\mathbb{N}} U_{p(n)}$ by well-definedness of Stabilize from (4.).

Finally, concatenating $u \mapsto \left(U_{u(1)} \cup \ldots \cup U_{u(|u|)}\right) : \mathbb{N}^* \to \mathcal{K}(\mathbf{X})$ with the ∇-computable inverse to $u \mapsto \left(U_{u(1)} \cup \ldots \cup U_{u(|u|)}\right) : \mathbb{N}^* \to \mathcal{O}(\mathbf{X})$ yields $\mathrm{id}_{\mathcal{O},\mathcal{K}} : \mathcal{O}(\mathbf{X}) \to (\mathcal{K}(\mathbf{X}))^{\nabla}$ as a well-defined and computable map. ◀

▶ **Corollary 37.** *Every Noetherian Quasi-Polish space is ∇-computably Noetherian relative to some oracle.*

Proof. It is clear that every Quasi-Polish space has a nice countably basis relative to some oracle, as we can just code all relevant information on the intersection of basis elements into the oracle. Then we use the 6th characterization from Theorem 36, and notice that the space being Noetherian suffices to prove the surjectivity of the map, and that access to the oracle making the countable basis nice suffices for the computability of the inverse. ◀

We point out some further simple observations on ∇-computably Noetherian spaces: All finite spaces containing only computable points are ∇-computably Noetherian.∇-computably Noetherian spaces are closed under finite products and finite coproducts, and computable images of ∇-computably Noetherian spaces are ∇-computably Noetherian. With some more

effort, one can also show that any computable well-quasiorder induces a ∇-computably Noetherian space (see [8]).

5 Noetherian spaces as ∇-compact spaces

For some hyperspace $P(\mathbf{X})$ of subsets of a represented space $\mathbf{X}$, and a space B of truth values $\bot$, $\top$, we define the map $\mathrm{isFull} : P(X) \to B$ by $\mathrm{isFull}(X) = \top$ and $\mathrm{isFull}(A) = \bot$ for $A \neq X$. We recall from [29] that a represented space is (computably) compact iff $\mathrm{isFull} : \mathcal{O}(\mathbf{X}) \to \mathbb{S}$ is continuous (computable).

The space $\mathbb{S}^\nabla \cong \mathbf{2}^\nabla$ can be considered as the space of $\mathbf{\Delta}^0_2$-truth values. In particular, we can identify $\mathbf{\Delta}^0_2$-subsets of $\mathbf{X}$ with their continuous characteristic functions into $\mathbf{2}^\nabla$, just as the open subsets are identifiable with their continuous characteristic functions into $\mathbb{S}$. By replacing both occurrences of $\mathbb{S}$ in the definition of compactness (one is hidden inside $\mathcal{O}$) by $\mathbb{S}^\nabla$, we arrive at:

▶ **Definition 38.** A represented space $\mathbf{X}$ is called ∇-compact, iff $\mathrm{isFull} : \mathbf{\Delta}^0_2(\mathbf{X}) \to \mathbb{S}^\nabla$ is computable.

▶ **Theorem 39.** *A Quasi-Polish space is ∇-compact iff it is ∇-computably Noetherian (relative to some oracle).*

The proof is provided in the following lemmata and propositions.

Recall that constructible subsets of a topological space are finite boolean combinations of open subsets. For a represented space $\mathbf{X}$, there is an obvious represented space $\mathfrak{C}(\mathbf{X})$ of constructible subsets of $\mathbf{X}$: A set $A \in \mathfrak{C}(\mathbf{X})$ is given by a (Goedel-number of a) boolean expression ϕ in n variables, and an n-tuple of open sets $U_1, \dots, U_n$ such that $A = \phi(U_1, \dots, U_n)$. Straight-forward calculation shows that we can always assume that $\phi(x_1, \dots, x_{2n}) = (x_1 \setminus x_2) \cup \dots \cup (x_{2n-1} \setminus x_{2n})$ without limitation of generality.

▶ **Lemma 40** ([4])**.** *Let $\mathbf{X}$ be a ∇-computably Noetherian Quasi-Polish space. Then* $\mathrm{id} : \mathbf{\Delta}^0_2(\mathbf{X}) \to \mathfrak{C}(\mathbf{X})^\nabla$ *is well-defined and computable.*

Proof. As $\mathbf{X}$ is Quasi-Polish, we can take it to be represented by an effectively open representation $\delta_{\mathbf{X}} : \mathbb{N}^\mathbb{N} \to \mathbf{X}$.

We can consider our input $A \in \mathbf{\Delta}^0_2(\mathbf{X})$ to be given by a realizer $f : \mathbb{N}^\mathbb{N} \to \{0,1\}$ of a finite mindchange computation. We consider the positions where a mindchange happens, i.e. those $w \in \mathbb{N}^*$ which if read by f will cause a mindchange to happen before reading any more of the input. W.l.o.g. we may assume that the realizer makes at most one mindchange at a given position $w \in \mathbb{N}^*$, and the realizer initially outputs 0 before reading any of the input.

Let $W \subseteq \mathbb{N}^*$ be the set of mindchange positions. To simplify the following, we will view ε (the empty string in $\mathbb{N}^*$) as being an element of W (this assumption can be justified formally by viewing the initial output of 0 as being a mindchange from "undefined" to 0). Note that W is decidable by simply observing the computation of f. If we denote the prefix relation on $\mathbb{N}^*$ by $\sqsubseteq$, we see that there are no infinite strictly ascending sequences in W with respect to $\sqsubseteq$, since any such sequence would correspond to an input that induces infinitely many mindchanges. It follows that $(W, \preceq)$ is a computable total well-order with maximal element ε, where $\preceq$ is the (restriction of the) Kleene-Brouwer order and defined as $v \preceq w$ if and only

[4] This is based on an adaption of the proof of the computable Hausdorff-Kuratowski theorem in [28].

if (i) $w \sqsubseteq v$, or (ii) $v(n) < w(n)$, where n is the least position where v and w are both defined and disagree.

We first note that $\min :\subseteq \mathbf{\Delta}^0_2(W) \to W$ is ∇-computable, where min is the function mapping each non-empty $S \in \mathbf{\Delta}^0_2(W)$ to the $\preceq$-minimal element of S. A realizer for min on input S can test in parallel whether each element of W is in S, and output as a guess the $\preceq$-minimal element which it currently believes to be in S. Since $\preceq$ is a well-order and it only takes finitely many mindchanges to determine whether or not a given element is in S, this computation is guaranteed to converge to the correct answer.

For each $w \in W$, define $U_w := \bigcup_{v \in W, v \preceq w} \delta_{\mathbf{X}}[v\mathbb{N}^{\mathbb{N}}]$, which is an effectively open subset of $\mathbf{X}$ and a uniform definition because $\preceq$ is decidable. Next, let $\mathbf{1} = \{*\}$ be the totally represented space with a single point, and define $h\colon W \to (\mathbf{\Delta}^0_2(W)+\mathbf{1})$ as $h(w) = *$ if $U_w = \mathbf{X}$ and $h(w) = \{v \in W \mid U_v \subsetneq U_w\}$, otherwise. The computability of the mapping $w \mapsto U_w$ and the assumption that $\mathbf{X}$ is ∇-computably Noetherian implies that it is ∇-decidable whether $U_w = \mathbf{X}$, and also that the characteristic function of the set $\{v \in W \mid U_v \subsetneq U_w\}$ is ∇-computable given $w \in W$. It follows that h is well-defined and ∇-computable.

We construct a finite sequence $v_0 \prec \ldots \prec v_k$ in W by defining $v_0 = \min(W)$ and $v_{n+1} = \min(h(v_n))$ whenever $h(v_n) \neq *$. This sequence is necessarily finite because the U_{v_n} form a strictly increasing sequence of open sets and $\mathbf{X}$ is Noetherian. Note that the last element v_k in the sequence satisfies $h(v_k) = *$. It follows that the sequence $\langle v_0, \ldots, v_k\rangle \in W^*$ can be ∇-computed from the realizer f because it only involves a finite composition of ∇-computable functions, and it can be ∇-decided when the sequence terminates.

Define $\eta\colon W \to \{0,1\}$ to be the computable function mapping each $w \in W$ to the output of the realizer f after the mindchange upon reading w (thus $\eta(\varepsilon) = 0$). For $n \leq k$ define $V_n := U_{v_n} \setminus \bigcup_{m<n} U_{v_m}$. We claim that $A = \bigcup\{V_n \mid 0 \leq n \leq k \;\&\; \eta(v_n) = 1\}$, from which it will follow that we can ∇-compute a name for $A \in \mathfrak{C}(\mathbf{X})$ from the realizer f.

Fix $x \in \mathbf{X}$, and let $w \in W$ be $\preceq$-minimal such that $x \in \delta_{\mathbf{X}}[w\mathbb{N}^{\mathbb{N}}]$. It follows that $x \in A$ if and only if $\eta(w) = 1$, because w is a prefix of some name p for x, and the $\preceq$-minimality of w implies that the realizer f does not make any additional mindchanges on input p after reading w. Next, let $n \in \{0, \ldots, k\}$ be the least number satisfying $x \in V_n$. It is clear that $w \preceq v_n$. Conversely, if $n = 0$ then $v_n = v_0 \preceq w$ by the $\preceq$-minimality of v_0. If $n > 0$, then $w \not\preceq v_{n-1}$ hence x is a witness to $U_{v_{n-1}} \subsetneq U_w$, which implies $v_n = h(v_{n-1}) \preceq w$. Thus $w = v_n$, and it follows that $x \in A$ if and only if $x \in \bigcup\{V_n \mid 0 \leq n \leq k \;\&\; \eta(v_n) = 1\}$, which completes the proof. ◀

▶ **Proposition 41.** Let $\mathbf{X}$ be ∇-computably Noetherian. Then $\mathrm{isFull} : \mathfrak{C}(\mathbf{X}) \to \mathbf{2}^{\nabla}$ is computable.

Proof. It is well-known that the sets in $\mathfrak{C}(\mathbf{X})$ have a normal form $A = (U_0 \setminus V_0) \cup \ldots \cup (U_n \setminus V_n)$, and this is obtainable uniformly. Now $A = X$ iff $\forall I \subseteq \{0, \ldots, n\}$ $\left(\bigcap_{j \notin I} V_j\right) \subseteq \left(\bigcup_{i \in I} U_i\right)$. To see this, first note that the special case $I = \{0, \ldots, n\}$ yields $X = \bigcup_{i \in I} U_i$. Now consider for each $x \in X$ the statement for $I = \{i \mid x \notin V_i\}$.

In a ∇-computably Noetherian space, we can compute $\left(\bigcap_{j \notin I} V_j\right)$ as a compact set, and decide its inclusion in $\left(\bigcup_{i \in I} U_i\right)$ with finitely many mindchanges. Doing this for the finitely many choices of I is unproblematic, thus yielding the claim. ◀

▶ **Proposition 42.** Let $\mathbf{X}$ admit a partition $(A_n)_{n \in \mathbb{N}}$ into non-empty $\mathbf{\Delta}^0_2$-sets. Then $\mathbf{X}$ is not ∇-compact.

Proof. Given some $(t_i)_{i\in\mathbb{N}} \in (\mathbf{2}^\nabla)^\mathbb{N}$, we can compute the set $A := \{x \in \mathbf{X} \mid \exists n \in \mathbb{N}\ x \in A_n \wedge t_n = 1\} \in \mathbf{\Delta}^0_2(\mathbf{X})$. If $\mathbf{X}$ were ∇-compact, then applying $\mathrm{isFull} : \mathbf{\Delta}^0_2(\mathbf{X}) \to \mathbb{S}^\nabla \cong \mathbf{2}^\nabla$ to A would yield a computable realizer of $\bigwedge : (\mathbf{2}^\nabla)^\mathbb{N} \to \mathbf{2}^\nabla$. ◀

Proof of Theorem 39. By combining Lemma 40 and Proposition 41, we see that for a ∇-computably Noetherian quasi-Polish space $\mathbf{X}$ the map $\mathrm{isFull} : \mathbf{\Delta}^0_2(\mathbf{X}) \to \mathbb{S}^\nabla$ is computable, i.e. it is ∇-compact. Conversely, if $\mathbf{X}$ is not Noetherian, then by Lemma 11 there is a countably-infinite $\mathbf{\Delta}^0_2$-partition of $\mathbf{X}$, so by Proposition 42, it cannot be ∇-compact. ◀

The significance of ∇-compactness and Theorem 39 lies in the following proposition that supplies the desired quantifier-elimination result. The proof is a straight-forward adaption of the corresponding result for compact spaces and open predicates from [29] (recalled here as Propositions 20,21), which in turn has [11] and [24] as intellectual predecessors. Note that as $\neg : \mathbb{S}^\nabla \to \mathbb{S}^\nabla$ is computable, it follows that ∇-compactness and ∇-overtness coincide:

▶ **Proposition 43.** The following are equivalent for a represented space $\mathbf{X}$:

1. $\mathbf{X}$ is ∇-compact.
2. For any represented space $\mathbf{Y}$, the map $\forall : \mathbf{\Delta}^0_2(\mathbf{X} \times \mathbf{Y}) \to \mathbf{\Delta}^0_2(\mathbf{Y})$ mapping R to $\{y \in \mathbf{Y} \mid \forall x \in \mathbf{X}\ (x, y) \in R\}$ is computable.
3. For any represented space $\mathbf{Y}$, the map $\exists : \mathbf{\Delta}^0_2(\mathbf{X} \times \mathbf{Y}) \to \mathbf{\Delta}^0_2(\mathbf{Y})$ mapping R to $\{y \in \mathbf{Y} \mid \exists x \in \mathbf{X}\ (x, y) \in R\}$ is computable.

▶ **Corollary 44.** *A formula built from $\mathbf{\Delta}^0_2$-predicates, boolean operations and universal and existential quantification over Noetherian quasi-Polish spaces defines itself a $\mathbf{\Delta}^0_2$-predicate.*

▶ **Corollary 45.** *Let $\mathbf{X} = \mathbf{X}_0 \times \ldots \times \mathbf{X}_n$ be a Noetherian Quasi-Polish space. If a subset $U \subseteq \mathbf{X}_0$ is definable using a finite expression involving open predicates in $\mathbf{X}$, boolean operations, and existential and universal quantification, then U is definable using a finite expression involving open predicates in $\mathbf{X}_0$ and boolean operations.*

Acknowledgements. We are grateful to the participants of the Dagstuhl Seminar *Well-quasi orders in Computer Science*[5] for valuable discussions and inspiration. In particular we would like to thank Jean Goubault-Larrecq and Takayuki Kihara.

References

1 Verónica Becher and Serge Grigorieff. Borel and Hausdorff hierarchies in topological spaces of Choquet games and their effectivization. *Mathematical Structures in Computer Science*, 25(7):1490–1519, 2015.

2 Vasco Brattka. Effective Borel measurability and reducibility of functions. *Mathematical Logic Quarterly*, 51(1):19–44, 2005.

3 Vasco Brattka, Matthew de Brecht, and Arno Pauly. Closed choice and a uniform low basis theorem. *Annals of Pure and Applied Logic*, 163(8):968–1008, 2012.

4 Vasco Brattka, Guido Gherardi, and Alberto Marcone. The Bolzano-Weierstrass Theorem is the jump of Weak König's Lemma. *Annals of Pure and Applied Logic*, 163(6):623–625, 2012. also arXiv:1101.0792.

5 Vasco Brattka and Arno Pauly. Computation with advice. *Electronic Proceedings in Theoretical Computer Science*, 24, 2010. CCA 2010.

[5] `http://www.dagstuhl.de/16031`

6 Matthew de Brecht. Quasi-Polish spaces. *Annals of Pure and Applied Logic*, 164(3):354–381, 2013.

7 Matthew de Brecht. Levels of discontinuity, limit-computability, and jump operators. In Vasco Brattka, Hannes Diener, and Dieter Spreen, editors, *Logic, Computation, Hierarchies*, pages 79–108. de Gruyter, 2014. arXiv 1312.0697.

8 Matthew de Brecht and Arno Pauly. Noetherian quasi-polish spaces. arXiv 1607.07291, 2016.

9 Matthew de Brecht and A. Yamamoto. Mind change complexity of inferring unbounded unions of pattern languages from positive data. *Theoretical Computer Science*, 411:976–985, 2010.

10 Martín Escardó. Synthetic topology of datatypes and classical spaces. *Electronic Notes in Theoretical Computer Science*, 87, 2004.

11 Martín Escardó. Intersections of compactly many open sets are open, 2009.

12 J. M. G. Fell. A Hausdorff topology for the closed subsets of a locally compact non-Hausdorff space. *Proceedings of the American Mathematical Society*, 13:472–476, 1962.

13 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1):63–92, 2001.

14 Jean Goubault-Larrecq. On Noetherian spaces. In *22nd Annual IEEE Symposium on Logic in Computer Science*, pages 453–462, 2007.

15 Jean Goubault-Larrecq. Noetherian spaces in verification. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *37th International Colloquium on Automata, Languages and Programming (ICALP) 2010, Proceedings, Part II*, pages 2–21. Springer, 2010.

16 Jean Goubault-Larrecq. A constructive proof of the topological Kruskal theorem. In Krishnendu Chatterjee and Jirí Sgall, editors, *38th International Symposium on the Mathematical Foundations of Computer Science (MFCS)*, pages 22–41. Springer, 2013.

17 Jean Goubault-Larrecq. *Non-Hausdorff Topology and Domain Theory*. New Mathematical Monographs. Cambridge University Press, 2013.

18 Vassilios Gregoriades, Tamás Kispéter, and Arno Pauly. A comparison of concepts from computable analysis and effective descriptive set theory. *Mathematical Structures in Computer Science*, 2016.

19 Reinhold Heckmann. Spatiality of countably presentable locales (proved with the Baire category theorem). *Mathematical Structures in Computer Science*, 25(7):1607–1625, 2015.

20 Rudolf-Eberhard Hoffmann. On the sobrification reminder ${}^s x - x$. *Pacific Journal of Mathematics*, 83(1):145–156, 1979.

21 J. E. Jayne and C. A. Rogers. First level Borel functions and isomorphisms. *Journal de Mathématiques Pures et Appliquées*, 61:177–205, 1982.

22 Takayuki Kihara. Decomposing Borel functions using the Shore-Slaman join theorem. *Fundamenta Mathematicae*, 230, 2015. arXiv 1304.0698.

23 Luca Motto Ros and Brian Semmes. A new proof of a theorem of Jayne and Rogers. *Real Analysis Exchange*, 35(1):195–204, 2009.

24 Leopoldo Nachbin. Compact unions of closed subsets are closed and compact intersections of open subsets are open. *Portugaliae mathematica*, 49(4):403–409, 1992.

25 C. S. J. A. Nash-Williams. On well-quasi-ordering infinite trees. *Mathematical Proceedings of the Cambridge Philosophical Society*, 61(3):697–720, 1965.

26 Eike Neumann and Arno Pauly. A topological view on algebraic computations models. arXiv:1602.08004, 2016.

27 Hugo Nobrega and Arno Pauly. Game characterizations and lower cones in the weihrauch degrees. In Jarkko Kari, Florin Manea, and Ion Petre, editors, *Unveiling Dynamics and*

Complexity – Computability in Europe (CiE), Proceedings, volume 10307 of *LNCS*, pages 327–337. Springer, 2017.

28 Arno Pauly. Computability on the countable ordinals and the Hausdorff-Kuratowski theorem. arXiv 1501.00386, 2015.

29 Arno Pauly. On the topological aspects of the theory of represented spaces. *Computability*, 5(2):159–180, 2016.

30 Arno Pauly and Matthew de Brecht. Towards synthetic descriptive set theory: An instantiation with represented spaces. arXiv 1307.1850.

31 Arno Pauly and Matthew de Brecht. Non-deterministic computation and the Jayne Rogers theorem. *Electronic Proceedings in Theoretical Computer Science*, 143, 2014. DCM 2012.

32 Arno Pauly and Matthew de Brecht. Descriptive set theory in the category of represented spaces. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 438–449, 2015.

33 R. Rado. Partial well-ordering of sets of vectors. *Mathematika*, 1(2):89–95, 1954.

34 Matthias Schröder. *Admissible Representations for Continuous Computations*. PhD thesis, FernUniversität Hagen, 2002.

35 Matthias Schröder. Extended admissibility. *Theoretical Computer Science*, 284(2):519–538, 2002.

36 Victor L. Selivanov. Difference hierarchy in φ-spaces. *Algebra and Logic*, 43(4):238–248, 2004.

37 M. B. Smyth. Quasi-uniformities: Reconciling domains with metric spaces. In M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Language Semantics*, pages 236–253. Springer, 1988.

38 Paul Taylor. A lambda calculus for real analysis. *Journal of Logic & Analysis*, 2(5):1–115, 2010.

39 Klaus Weihrauch. The TTE-interpretation of three hierarchies of omniscience principles. Informatik Berichte 130, FernUniversität Hagen, Hagen, September 1992.

40 Klaus Weihrauch. *Computable Analysis*. Springer-Verlag, 2000.

41 Klaus Weihrauch. Computable separation in topology, from T_0 to T_2. *Journal of Universal Computer Science*, 16(18):2733–2753, 2010.

42 Martin Ziegler. Revising type-2 computation and degrees of discontinuity. *Electronic Notes in Theoretical Computer Science*, 167:255–274, 2007.

Fast(er) Reasoning in Interval Temporal Logic*

Davide Bresolin[1], Emilio Muñoz-Velasco[2], and Guido Sciavicco[3]

1 **Department of Mathematics, University of Padova, Italy**
`davide.bresolin@unipd.it`
2 **Department of Applied Mathematics, University of Málaga, Spain**
`ejmunoz@uma.es`
3 **Department of Mathematics and Computer Science, University of Ferrara, Italy**
`guido.sciavicco@unife.it`

Abstract

Clausal forms of logics are of great relevance in Artificial Intelligence, because they couple a high expressivity with a low complexity of reasoning problems. They have been studied for a wide range of classical, modal and temporal logics to obtain tractable fragments of intractable formalisms. In this paper we show that such restrictions can be exploited to lower the complexity of interval temporal logics as well. In particular, we show that for the Horn fragment of the interval logic $\mathbf{A}\overline{\mathbf{A}}$ (that is, the logic with the modal operators for Allen's relations *meets* and *met by*) without diamonds the complexity lowers from NExpTime-complete to P-complete. We prove also that the tractability of the Horn fragments of interval temporal logics is lost as soon as other interval temporal operators are added to $\mathbf{A}\overline{\mathbf{A}}$, in most of the cases.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Temporal Logic, Horn Fragments, Satisfiability, Complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.17

1 Introduction

Sub-propositional logics are particularly interesting from a practical point of view, because they couple a high expressivity with a low complexity of reasoning problems. There are two standard ways to weaken the classical propositional language based on the clausal form of formulas: the *Horn fragment*, that only allows clauses with at most one positive literal [18], and the *Krom fragment*, that only allows clauses with at most two (positive or negative) literals [20]. The *core fragment* combines both restrictions. In the case of *modal* logics two more possible restrictions arise, namely by excluding the use of *universal* or the use of *existential* modal operators, giving rise, respectively, to the *diamond fragment* and the *box fragment* of modal logic. Such restrictions apply, as well, to temporal, spatial, and description logic, and, in each case, combined with classical sub-propositional restriction, they generate a complex scenario encompassing several fragments.

Weakening a logic is motivated by the search of computationally well-behaved fragments. For example, the satisfiability problem for classical propositional Horn logic is P-complete, while the classical propositional Krom logic satisfiability problem (also known as the 2-SAT problem) is NLogSpace-complete [23], and the same holds for the core fragment, where

* This work was partially supported by the Spanish Research Project TIN15-70266-C2-P-1 (E. Muñoz-Velasco), and by the Italian INdAM-GNCS project *Logics and Automata for Interval Model Checking* (D. Bresolin and G. Sciavicco).

the classical NLogSpace-hardness theorem for 2-SAT can be easily strengthened to deal with the case of binary Horn clauses. The satisfiability problem for quantified propositional logic (**QBF**), which is PSpace-complete in its general form, becomes P when formulas are restricted to binary (Krom) clauses [7]. Horn fragments of modal logic **K** and of several of its axiomatic extensions have been considered in [12, 13, 15, 16, 22]; in particular, it is known that **K**, **T**, **K4**, and **S4** are all PSpace-complete even under the Horn restriction, but they become P-complete under the Horn restriction without diamonds [12]. In [4, 14], the authors study different sub-propositional fragments LTL. By excluding the operators Since and Until from the language, and keeping only the Next/Previous-time operators and the box version of Future and Past, it is possible to prove that the Krom and core fragments are NP-hard, while the Horn fragment is still PSpace-complete (the same complexity of the full language). Moreover, the complexity of the Horn, Krom, and core fragments without Next/Previous-time operators range from NLogSpace (core), to P (Horn), to NP-hard (Krom). Where only a *global* (anywhere in time) modality is allowed, their complexity is even lower (from NLogSpace to P). Temporal extensions of the description logic DL-Lite have been studied under similar sub-propositional restrictions, and similar improvements in the complexity have been obtained [5].

In this paper we show that clausal forms can be exploited to lower the complexity of interval temporal logics as well. Halpern and Shoham's Modal Logic of Allen's Relations (**HS**) [17] is a multi-modal logic where each world is interpreted as an *interval* in time, and accessibility relations are built over Allen's relations [2]. So, **HS** is the classical propositional logic extended with six modal operators *later* (denoted by $\langle L \rangle$), *meets* ($\langle A \rangle$), *overlaps* ($\langle O \rangle$), *during* ($\langle D \rangle$), *finishes* ($\langle E \rangle$), and *starts* ($\langle B \rangle$), plus their inverse ones (equality does not play any role in modal logic). The finite satisfiability problem for fragments of **HS** (that is, logics that emerge from a systematic exclusion of some of these modalities) has been studied in [9], resulting in a very informative picture of the various classes of fragments classified by their computational complexity. The Horn, Krom, and core restrictions of **HS** are still undecidable [11], but weaker restrictions have shown positive results. In particular, the Horn fragment of **HS** without diamonds becomes P-complete in two interesting cases [6, 8]: first, when it is interpreted over dense linear orders, and, second, when the semantics of its modalities becomes reflexive. On the other hand, in the classical irreflexive semantics, interpreted on finite/discrete linear orders, the Horn fragment of **HS** without diamonds is still undecidable, and this justifies our interest in finding well-behaved syntactical fragments of it. Without restrictions, the satisfiability for **A** and for $\mathbf{A}\overline{\mathbf{A}}$ is NExpTime-complete in the finite case, the case of natural numbers, the integers, and the class of all discrete linear orders [10]. The purpose of this paper is to prove that their Horn fragments without diamonds become P-complete at least in the finite case, which is (usually) emblematic for the behaviour of interval temporal logics in the whole range of classes of discrete linear orders. We shall also prove that tractability of these fragments is lost as soon as other modal operators from the **HS** machinery are added, in most of the cases.

2 Preliminaries

Let $\mathbb{D} = \langle D, < \rangle$ be a linearly ordered set. An *interval* over $\mathbb{D}$ is an ordered pair $[x, y]$, where $x, y \in D$ and $x < y$[1]. There are 12 different relations between two intervals in a linear order,

[1] This definition excludes point intervals $[x, x]$, and conforms to the one adopted by Allen in [1] and by most of the recent literature; it differs from the one in [17], where point intervals are allowed.

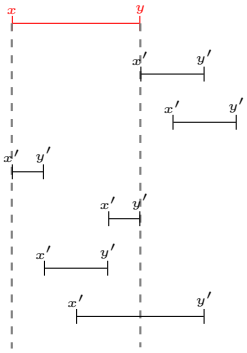
HS	Allen's relations	Graphical representation
$\langle A \rangle$	$[x,y]R_A[x',y'] \Leftrightarrow y = x'$	
$\langle L \rangle$	$[x,y]R_L[x',y'] \Leftrightarrow y < x'$	
$\langle B \rangle$	$[x,y]R_B[x',y'] \Leftrightarrow x = x', y' < y$	
$\langle E \rangle$	$[x,y]R_E[x',y'] \Leftrightarrow y = y', x < x'$	
$\langle D \rangle$	$[x,y]R_D[x',y'] \Leftrightarrow x < x', y' < y$	
$\langle O \rangle$	$[x,y]R_O[x',y'] \Leftrightarrow x < x' < y < y'$	

Figure 1 Allen's interval relations and **HS** modalities.

often called *Allen's relations* [1]: the six relations R_A (adjacent to), R_L (later than), R_B (begins), R_E (ends), R_D (during), and R_O (overlaps), depicted in Fig. 1, and their inverses $R_{\overline{X}} = (R_X)^{-1}$, for each $X \in \{A, L, B, E, D, O\}$.

Halpern and Shoham's logic **HS** [17] is a multi-modal logic with formulae built from a finite, non-empty set $\mathcal{AP}$ of atomic propositions (or proposition letters), the classical propositional connectives, and a modality for each Allen relation:

$$\varphi ::= \bot \mid p \mid \neg\psi \mid \psi \vee \xi \mid \psi \wedge \xi \mid \langle X \rangle \psi \mid \langle \overline{X} \rangle \psi,$$

where $p \in \mathcal{AP}$ and $X \in \{A, L, B, E, D, O\}$. The other propositional connectives and constants (e.g., $\rightarrow$, and $\top$), and the universal modalities $[X]$ and $[\overline{X}]$ can be derived in the standard way. A *syntactical fragment* of **HS** is any fragment obtained by selecting a specific subset $\mathcal{F} = \{X_1, X_2, \ldots, X_n\}$ of relations, and denoted by $\mathrm{X_1X_2 \ldots X_n}$.

The semantics of **HS** is given in terms of *interval models* $M = \langle \mathbb{I}(\mathbb{D}), V \rangle$, where $\mathbb{D} = \langle D, < \rangle$ is a linear order, $\mathbb{I}(\mathbb{D})$ is the set of all (strict) intervals over $\mathbb{D}$, and V is a *valuation function* $V : \mathcal{AP} \mapsto 2^{\mathbb{I}(\mathbb{D})}$, which assigns to each atomic proposition $p \in \mathcal{AP}$ the set of intervals $V(p)$ on which p holds. The *truth* of a formula φ on a given interval $[x, y]$ in an interval model M is defined by structural induction on formulae, as follows:

- $M, [x, y] \Vdash p$ if $[x, y] \in V(p)$, for $p \in \mathcal{AP}$;
- $M, [x, y] \Vdash \neg\psi$ if $M, [x, y] \nVdash \psi$;
- $M, [x, y] \Vdash \psi \vee \xi$ if $M, [x, y] \Vdash \psi$ or $M, [x, y] \Vdash \xi$;
- $M, [x, y] \Vdash \psi \wedge \xi$ if $M, [x, y] \Vdash \psi$ and $M, [x, y] \Vdash \xi$;
- $M, [x, y] \Vdash \langle X \rangle \psi$ if there exists $[z, t]$ such that $[x, y]R_X[z, t]$ and that $M, [z, t] \Vdash \psi$;
- $M, [x, y] \Vdash \langle \overline{X} \rangle \psi$ if there exists $[z, t]$ such that $[x, y]R_{\overline{X}}[z, t]$ and that $M, [z, t] \Vdash \psi$.

In this paper we are interested in finite models, so that for all intent and purposes, we assume that $D = \{0, 1, \ldots, N - 1\}$ is a finite domain. If a formula φ is satisfiable on a model of length N, we say that φ is *N-satisfiable*. The finite satisfiability problems of **HS** is undecidable, and the computational complexity of its decidable (syntactical) fragments range from P-complete to non-primitive recursive (see [9] and references within).

In order to define sub-propositional fragments of **HS** we start from the *clausal form* of formulas of **HS**, whose building blocks are the *positive literals*:

$$\lambda \quad ::= \quad \top \mid p \mid \langle X \rangle \lambda \mid [X] \lambda \mid \langle \overline{X} \rangle \lambda \mid [\overline{X}] \lambda, \tag{1}$$

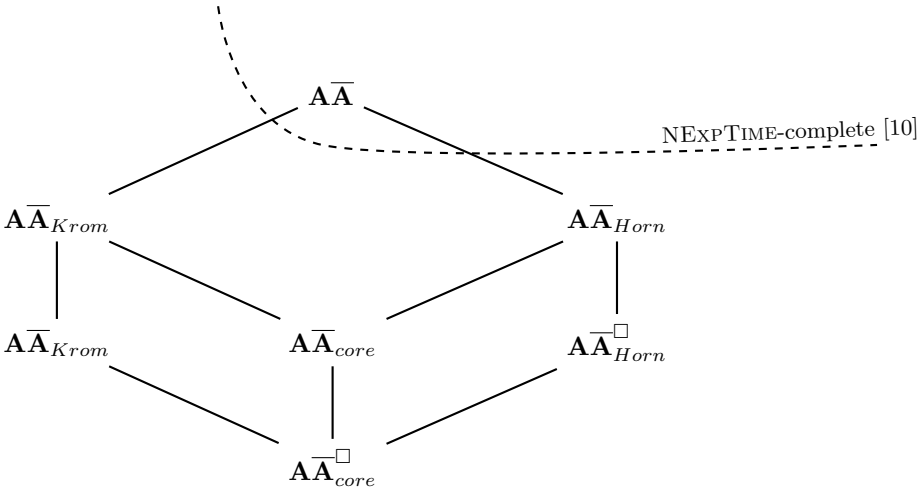


Figure 2 Sub-propositional fragments of $\mathbf{A\overline{A}}$.

and we say that φ is in *clausal form* if it can be generated by the following grammar:

$$\varphi \quad ::= \quad \lambda \mid \psi \wedge \xi \mid [U](\neg\lambda_1 \vee \ldots \vee \neg\lambda_n \vee \lambda_{n+1} \vee \ldots \vee \lambda_{n+m}), \tag{2}$$

where $[U](\neg\lambda_1 \vee \ldots \vee \neg\lambda_n \vee \lambda_{n+1} \vee \ldots \vee \lambda_{n+m})$ is a *clause*, and $[U]$ is the *global operator*. A formula $[U]\psi$ is true on the interval $[x, y]$ if and only if ψ holds on every interval $[z, t]$ of the model. Depending on the language, the global operator can be either definable using the other modalities, or included as a primitive modality. From now on we follow the latter approach and we consider $[U]$ as part of the language. In the following, we use $\varphi_1, \varphi_2, \ldots$ to denote clauses, $\lambda_1, \lambda_2, \ldots$ to denote positive literals, and $\varphi, \psi, \xi, \ldots$ to denote formulas. We write clauses in their implicative form $[U](\lambda_1 \wedge \ldots \wedge \lambda_n \rightarrow \lambda_{n+1} \vee \ldots \vee \lambda_{n+m})$, and use $\bot$ as a shortcut for $\neg\top$. It is known that every modal logic formula can be rewritten in clausal form [22], and this holds for **HS**, as well. Thus, any formula φ can be written as a conjunction $\varphi_1 \wedge \ldots \wedge \varphi_n$ of positive literals and global clauses and, from now on, will be represented as the set $\{\varphi_1, \ldots, \varphi_n\}$ of positive literals and global clauses.

Sub-propositional fragments of **HS** can be defined by constraining the cardinality and the structure of clauses: the fragment in which each clause in (2) is such that $m \leq 1$ is called *Horn* fragment, and denoted by $\mathbf{HS}_{Horn}$, while the fragment in which each clause is such that $n + m \leq 2$ it is called *Krom* fragment, and it is denoted by $\mathbf{HS}_{Krom}$; when both restrictions apply we obtain the *core* fragment, denoted by $\mathbf{HS}_{core}$. We constrain fragments also by allowing only the use of universal modalities and only existential modalities in positive literals, obtaining the *box* and *diamond* fragments of modal logic **HS**. In this way, we define $\mathbf{HS}^{\Box}_{Horn}$ and $\mathbf{HS}^{\Diamond}_{Horn}$ as, respectively, the box and the diamond fragments of $\mathbf{HS}_{Horn}$. By applying the same restrictions to $\mathbf{HS}_{Krom}$ and $\mathbf{HS}_{core}$, we obtain the pair $\mathbf{HS}^{\Box}_{Krom}$ and $\mathbf{HS}^{\Diamond}_{Krom}$ from the former, and the pair $\mathbf{HS}^{\Box}_{core}$ and $\mathbf{HS}^{\Diamond}_{core}$, from the latter. These definitions are borrowed from [13, 12]. In this paper we are interested in studying sub-propositional syntactical fragments of **HS**, and, in particular, the fragment $\mathbf{A\overline{A}}$ under the Horn restriction without diamonds, that is, $\mathbf{A\overline{A}}^{\Box}_{Horn}$; the interesting fragments are shown in Fig. 2, ordered by syntactical containment.

The fragment $\mathbf{A\overline{A}}^{\Box}_{Horn}$ is very similar to its reflexive version (see [6, 8]) but there are significant differences. For instance, in the finite/discrete case the satisfiability problem for irreflexive $\mathbf{HS}^{\Box}_{Horn}$ is still undecidable, while it is P-complete in its reflexive version,

which means that, in general, the good computational behaviour of a fragment of **HS** is not necessarily related to that of its reflexive variant. As we have already pointed out, in this paper we limit ourselves to study the complexity of the satisfiability problem in the finite case: this is not only interesting on its own, but, also, it is emblematic for the entire range of (strongly) discrete linearly ordered sets, such as $\mathbb{N}$ or $\mathbb{Z}$.

Finally, observe that the fragment $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, despite its weakness, can be still used to describe meaningful situations. First, notice that $[U](\langle A\rangle\lambda \wedge \varphi \rightarrow \lambda')$ can be written as $[U](\lambda \rightarrow [\overline{A}](\varphi \rightarrow \lambda'))$, which is to say that $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$ allows a (limited) use of diamonds. Then, for example, we can express a medical statement such as *During therapy no patient is allowed to smoke* with a simple combination of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$:

$$\begin{array}{ll} [U](\mathit{therapy} \rightarrow [A](\mathit{stop} \wedge [A]\mathit{stop})) & \mathit{therapy's\ boundaries} \\ [U](\mathit{therapy} \rightarrow [\overline{A}][A](\langle A\rangle\langle A\rangle(\mathit{smoking} \wedge \langle A\rangle\mathit{stop}) \rightarrow \bot)) & \left.\begin{array}{l}\\\end{array}\right\} \ \mathit{smoking\ constraint} \\ [U](\mathit{therapy} \rightarrow [\overline{A}][A](\langle A\rangle(\mathit{smoking} \wedge \langle A\rangle\mathit{stop}) \rightarrow \bot)) & \end{array}$$

3 P-Completeness of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$

The complexity of the finite satisfiability problem for $\mathbf{A}\overline{\mathbf{A}}$ is NExpTime-complete. In this section, we prove that the same problem for $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$ is P-complete (and, therefore, for $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{core}$ it is in P), and hence that, at least in this case, interval logics are showing a similar behaviour to modal logic [12].

The fragment of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$ in which we limit the modal depth to one is *model-extending equivalent* to $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, that is, equivalent at the price of adding fresh propositional letters, with a polynomial translation that witnesses such an equivalence. Therefore, for the purposes of this section, we may assume that every literal has modal depth at most one. Given a finitely satisfiable set of clauses φ, among any set of models of fixed length N, we can define a notion of *ordering*: we say that $M_1 \leq_N M_2$ if and only if M_1 and M_2 are based on the same domain (of length N) and, for each $p \in \mathcal{AP}$, $V_1(p) \subseteq V_2(p)$; obviously, $M_1 <_N M_2$ if $M_1 \leq_N M_2$ and $M_1 \neq M_2$.

▶ **Lemma 1.** *Let φ be a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, and let $\mathcal{M}$ be the (finite) set of models of length N that satisfy φ on the interval $[x, y]$. Then, $(\mathcal{M}, \leq_N)$ has a minimum.*

Proof. Let φ be a satisfiable set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, and let $\mathcal{M}$ be the set of interval models of length N that satisfy φ. Obviously, $\mathcal{M}$ is finite. If $\mathcal{M}$ is a singleton, then the property is trivially true. Suppose, then, that there exist at least two $\leq_N$-incomparable interval models, denoted by M_1 and M_2, in $\mathcal{M}$. Thus, $M_1, [x, y] \Vdash \varphi$ and $M_2, [x, y] \Vdash \varphi$. We define a new model $M' = \langle \mathbb{I}(\mathbb{D}), V'\rangle$ based on the same domain as M_1 and M_2 and such that its valuation V', for each propositional letter p, is $\{[t, t'] \mid [t, t'] \in V_1(p) \cap V_2(p)\}$. We claim that $M', [x, y] \Vdash \varphi$. Suppose, by way of contradiction, that this is not the case. So, there exists a clause φ_i of φ such that $M', [x, y] \nVdash \varphi_i$. Several cases arise:

- If $\varphi_i = p$ for some propositional letter p, then we have a contradiction given that $M', [x, y] \nVdash \varphi_i$ implies that $M_1, [x, y] \nVdash p$ or $M_2, [x, y] \nVdash p$;
- If $\varphi_i = [A]p$ (the case for $\varphi_i = [\overline{A}]p$ is symmetrical) for some propositional letter p, then we have a contradiction given that $M', [x, y] \nVdash \varphi_i$ implies that for some $z > y$ $M', [y, z] \nVdash p$, that is, $M_1, [y, z] \nVdash p$ or $M_2, [y, z] \nVdash p$, that is, $M_1, [x, y] \nVdash [A]p$ or $M_2, [x, y] \nVdash [A]p$;

- If $\varphi_i = [U](\lambda_1 \wedge \ldots \wedge \lambda_n \rightarrow \lambda)$ or $\varphi_i = [U](\lambda_1 \wedge \ldots \wedge \lambda_n \rightarrow \bot)$, then, for some $[t,t']$, we have that $M', [t,t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$ and $M', [t,t'] \nVdash \lambda$. Consider the generic antecedent λ_j. If it is a propositional letter p, then $M', [t,t'] \Vdash \lambda_j$ implies that both $M_1, [t,t'] \Vdash \lambda_j$ and $M_2, [t,t'] \Vdash \lambda_j$. On the other hand, if $\lambda_j = [A]p$ (the case for $[\overline{A}]p$ is symmetrical) for some propositional letter p, then $M', [t,t'] \Vdash \lambda_j$ implies that, for each $t'' > t'$, $M', [t',t''] \Vdash p$, which, in turn, implies that both $M_1, [t',t''] \Vdash p$ and $M_2, [t',t''] \Vdash p$ for each $t'' > t'$. Either way, if $M', [t,t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$, then both $M_1, [t,t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$ and $M_2, [t,t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$. If $\lambda = \bot$, this is already a contradiction. If $\lambda = p$ for some propositional letter p, then both $M_1, [t,t'] \Vdash p$ and $M_2, [t,t'] \Vdash p$, which implies that $M', [t,t'] \Vdash p$, which is a contradiction. If $\lambda = [A]p$ (and, again, the case for $[\overline{A}]p$ is symmetrical) for some propositional letter p, then we have that both $M_1, [t',t''] \Vdash p$ and $M_2, [t',t''] \Vdash p$ for each $t'' > t$, that is, $M', [t,t'] \Vdash [A]p$, which is a contradiction.

This proves that M' is an interval model such that $M', [x,y] \Vdash \varphi$, that is, that $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$ is closed under intersection. Clearly, $M' \leq_N M_1, M_2$ and $M' \in \mathcal{M}$: by iterating this process we prove that $\mathcal{M}$ has a minimum with respect to $\leq_N$. ◀

Notice that the previous result strongly depends on the absence of diamonds in the language.

The above results prove that the finite satisfiability of a set φ of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$ can be reduced to the problem of finding an interval model M of length N that satisfies φ on some $[x,y]$ and is the minimum among the models of length N that satisfy φ. Let us call such a model an *N-minimum model* (or, simply, *minimum model*). Now, we want to prove that every set φ of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$ is finitely satisfiable if and only if it is satisfiable on a minimum model of cardinality polynomial in $|\varphi|$.

For a set φ of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, let $\mathcal{L}(\varphi)$ be the *literal closure* of φ, that is, the set of all the positive literals that occur as subformulas of φ. Given a model M we can univocally identify the literals that are satisfied on each interval $[z,t]$, and we can define the *label* of $[z,t]$ as $L([z,t]) = \{\lambda \in \mathcal{L} \mid M, [z,t] \Vdash \lambda\}$. Positive literals of the type $[A]p$ (resp., $[\overline{A}]p$) are called *A-temporal* literals (resp., *$\overline{A}$-temporal* literals). It is easy to observe that any two intervals $[z,t]$ and $[z',t]$ that end at the same point should satisfy the same set of A-temporal literals. Thus, it make sense to define the set of *A-requests* of a point t in the model M as the set $A(t)$ of all literals of the type $[A]p$ that occur in labels of intervals that end at t. Symmetrically, one can define the set of *$\overline{A}$-requests* (denoted $\overline{A}(t)$), and, in fact, associate each point t of M with a unique pair $(\overline{A}(t), A(t))$. As we shall see, requests in a minimum model should behave in a regular way: if $M, [x,y] \Vdash \varphi$, we say that M is *standard* if and only if, for each $z \neq x, y$, $A(z) \subseteq A(z+1)$ and $\overline{A}(z+1) \subseteq \overline{A}(z)$.

There are two key ingredients to prove that the maximal dimension of a minimum model is polynomial: first, we show that minimum models are necessarily standard, and, then, that the length of minimum standard models can be bounded to a polynomial number in $|\varphi|$.

▶ **Lemma 2.** *Let φ be a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, and let M be a N-minimum model that satisfies it. Then, M is standard.*

Proof. Let φ be a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, and let $M = \langle \mathbb{I}(\mathbb{D}), V \rangle$ be a model that satisfies it, that is, $M, [x,y] \Vdash \varphi$; we proceed by contradiction, and we prove that if M is not standard, then it cannot be minimum. Without loss of generality, let $z > y$ be a point in M such that $A(z) \not\subseteq A(z+1)$. This means that for some $[A]p$, it is the case that $[A]p \in A(z)$ and $[A]p \notin A(z+1)$. We define a new model $M' = \langle \mathbb{I}(\mathbb{D}), V' \rangle$ based on the same domain as M and such that its valuation V', for each propositional letter p, is the union of:

1. $\{[t,t'] \mid [t,t'] \in V(p)$ and $t, t' \neq z\}$,

2. $\{[t,z] \mid t < z$ and $[t,z],[t,z+1] \in V(p)$ for each $t < z\}$,
3. $\{[z,z+1] \mid [z,z+1] \in V(p)\}$, and
4. $\{[z,t'] \mid t' > z+1$ and $[z,t'],[z+1,t'] \in V(p)$ for each $t' > z\}$.

We claim that $M', [x,y] \Vdash \varphi$. Suppose, by way of contradiction, that this is not the case. So, there exists a clause φ_i of φ such that $M', [x,y] \nVdash \varphi_i$. Several cases arise:

- If $\varphi_i = p$ for some propositional letter p, then we have a contradiction given that $M, [x,y] \Vdash \varphi_i$ and M and M' agree on the label of $[x,y]$, as per point (1) of the construction;
- If $\varphi_i = [\overline{A}]p$ for some propositional letter p, then we have a contradiction given that $M, [x,y] \Vdash \varphi_i$ and M and M' agree on the label of every interval of the type $[t,x]$, as per point 1 of the construction, because $t, x \neq z$;
- If $\varphi_i = [A]p$ for some propositional letter p, then, since $M, [x,y] \Vdash \varphi_i$, we know that, in M, for each $t > y$, $M, [y,t] \Vdash p$. By construction (points (1) and (2)), $M', [y,t] \Vdash p$, which implies that $M', [x,y] \Vdash [A]p$, leading to a contradiction;
- If $\varphi_i = [U](\lambda_1 \wedge \ldots \wedge \lambda_n \rightarrow \lambda)$ or $\varphi_i = [U](\lambda_1 \wedge \ldots \wedge \lambda_n \rightarrow \bot)$, then, for some $[t,t']$, we have that $M', [t,t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$ and $M', [t,t'] \nVdash \lambda$. There are several sub-cases:
  - Assume, first, that $t, t' < z$, and consider the generic antecedent λ_j. If it is a propositional letter p, then $M', [t,t'] \Vdash \lambda_j$ implies that $M, [t,t'] \Vdash \lambda_j$. On the other hand, if $\lambda_j = [A]p$ for some propositional letter p, then $M', [t,t'] \Vdash \lambda_j$ implies that, for each $t'' > t'$ (including $t'' = z$), $M', [t',t''] \Vdash p$, which in turn, by point (1) and (2) of the construction, implies that $M, [t',t''] \Vdash p$ for each $t'' > t'$. If $\lambda_j = [\overline{A}]p$ for some propositional letter p, then $M', [t,t'] \Vdash \lambda_j$ means that, for each $t'' < t$, $M', [t'',t] \Vdash p$, which implies, by point (1), that $M, [t'',t] \Vdash p$ for each $t'' < t$. Either way, if $M', [t,t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$, then $M, [t,t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$. If $\lambda = \bot$, this is already a contradiction. If $\lambda = p$ for some propositional letter p, then the contradiction is, again, immediate thanks to point (1) of the construction. Now, if $\lambda = [A]p$ for some propositional letter p, then we have that $M', [t',t''] \nVdash p$ for some $t'' > t'$; if $t'' \neq z$, then the label of $[t',t'']$ is preserved from M to M', which means that $M, [t',t''] \nVdash p$, leading to a contradiction, and, if $t'' = z$, then we have a contradiction with the fact that the label of $[t',z]$ in M' is the intersection of the labels of $[t',z]$ and $[t',z+1]$ in M, due to point (2) of the construction. If, finally, $\lambda = [\overline{A}]p$ for some propositional letter p, then we have that $M', [t'',t'] \nVdash p$ for some $t'' < t'$, which is a contradiction again with the fact that the label of each interval of the type $[t'',t']$ $(t' < z)$ is preserved from M to M';
  - Suppose, now, that $t' = z$ and $t < t'$. If $\lambda_j = p$ for some propositional letter p, then $M', [t,z] \Vdash p$ implies, by point (2), that $M, [t,z] \Vdash p$ and $M, [t,z+1] \Vdash p$. If $\lambda_j = [A]p$ for some propositional letter p, then $M', [t,z] \Vdash [A]p$ implies that $M', [z,t'] \Vdash p$ for each $t' > z$. But, this implies, by points (3) and (4), that $M, [z,t'] \Vdash p$ and, if $t' > z+1$, also that $M, [z+1,t'] \Vdash p$, which, in turn, implies that $M, [t,z] \Vdash [A]p$ and $M, [t,z+1] \Vdash [A]p$. Finally, if $\lambda_j = [\overline{A}]p$ for some propositional letter p, $M', [t,z] \Vdash [\overline{A}]p$ immediately implies that $M, [t,z] \Vdash [\overline{A}]p$ and $M, [t,z+1] \Vdash [\overline{A}]p$. Therefore, $M', [t,z] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$ implies that $M, [t,z] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$ and $M, [t,z+1] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$. If $\lambda = \bot$, this is already a contradiction. Otherwise, we have that $M, [t,z] \Vdash \lambda$ and $M, [t,z+1] \Vdash \lambda$. If $\lambda = p$ for some propositional letter p, then $M', [t,z] \nVdash p$ is a contradiction, since the label of $[t,z]$ in M' is the intersection of the labels of $[t,z]$ and $[t,z+1]$ in M. If $\lambda = [A]p$ for some propositional letter p, that is, if $M', [z,t''] \nVdash p$ for some $t'' > z$, we have a contradiction with the fact that, for each $t'' > z$, the label of $[z,t'']$ in M' is the intersection of the labels of $[z,t'']$ and $[z+1,t'']$ in M. If, finally, $\lambda = [\overline{A}]p$ for some

propositional letter p, then we have that $M', [t'', t] \Vdash \neg p$ for some $t'' < t$, which is a contradiction with the fact that the label of each interval of the type $[t'', t]$ $(t < z)$ is preserved from M to M';

- If $t < z$ and $t' \geq z + 1$, then the contradiction is immediate, as the labels of every interval that may possibly be involved is preserved from M to M';
- Suppose, now, that $t = z$ and $t' \geq z + 1$. If $\lambda_j = p$ for some propositional letter p, then $M', [z, t'] \Vdash p$ implies that $M, [z, t'] \Vdash p$ and (if $t' > z + 1$) $M, [z + 1, t'] \Vdash p$. If $\lambda_j = [A]p$ for some propositional letter p, $M', [z, t'] \Vdash [A]p$ immediately implies that $M, [z, t'] \Vdash [A]p$ and (if $t' > z + 1$) $M, [z + 1, t'] \Vdash [A]p$, since the labels of intervals that start at $t' \geq z + 1$ are preserved from M to M'. Finally, if $\lambda_j = [\overline{A}]p$ for some propositional letter p, $M', [z, t'] \Vdash [\overline{A}]p$ implies that $M', [t'', z] \Vdash p$ for each $t'' < z$; this, in turn, implies that for each $t'' < z$ we have that $M, [t'', z] \Vdash p$ and $M, [t'', z + 1] \Vdash p$, that is, that $M, [z, t'] \Vdash [\overline{A}]p$ and (if $t' > z + 1$) $M, [z + 1, t'] \Vdash [\overline{A}]p$. Therefore, $M', [z, t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$ implies that $M, [z, t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$ and (if $t' > z + 1$) $M, [z + 1, t'] \Vdash \lambda_1 \wedge \ldots \wedge \lambda_n$. If $\lambda = \bot$, this is already a contradiction. Otherwise, we have that $M, [z, t'] \Vdash \lambda$ and (if $t' > z + 1$) $M, [z + 1, t'] \Vdash \lambda$. If $\lambda = p$ for some propositional letter p, then $M', [z, t'] \nVdash p$ is a contradiction, since the label of $[z, t']$ in M' is preserved from M if $t' = z + 1$ and it is the intersection of the labels of $[z, t']$ and $[z + 1, t']$ in M otherwise. If $\lambda = [A]p$ for some propositional letter p, that is, if $M', [z, t'] \nVdash p$ for some $t' > z$, we have a contradiction with the fact that, for each $t'' > t'$, the label of $[t'', t']$ is preserved from M to M'. If, finally, $\lambda = [\overline{A}]p$ for some propositional letter p, then we have that, in M', $M', [t'', z] \Vdash \neg p$ for some $t'' < z$, that is $M, [t'', z] \Vdash \neg p$ or (if $t > z + 1$) $M, [t'', z + 1] \Vdash \neg p$, which is in contradiction with the fact that both $M, [z, t] \Vdash [\overline{A}]p$ and (if $t > z + 1$) $M, [z + 1, t] \Vdash [\overline{A}]p$;
- If $t \geq z + 1$, then the contradiction is immediate, as the labels of every interval that may possibly be involved are preserved from M to M'.

Thus, $M', [x, y] \Vdash \varphi$. Now, observe that by hypothesis, in M, $[A]p \in A(z)$ and $[A]p \notin A(z+1)$ for some propositional letter p. Then $M' <_N M$, since, at the very least, p is true in every interval of M starting at z while in M' it is false for some interval starting at z (because in M it is false at some interval starting at $z + 1$). Therefore, M cannot be N-minimum, as we wanted to show. The case in which $z < y$ (recall that $z \neq x$), as well as the case in which M is not standard because of some literal of the type $[\overline{A}]p$, can be treated in a very similar way. ◀

Given a standard model, we say that each $[A]p \in A(z + 1) \setminus A(z)$ (resp., $[\overline{A}]p \in \overline{A}(z) \setminus \overline{A}(z + 1)$), for a generic point z, is a *blocking* temporal literal.

▶ **Lemma 3.** *Let φ be a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, and let M be the N-minimum standard model that satisfies it. If $N \geq 6 \cdot |\varphi| + 3$, then there exists a minimum standard model M' of length $N' < N$ that satisfies φ.*

Proof. Let φ be a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, and let $M = \langle \mathbb{I}(\mathbb{D}), V \rangle$ be the N-minimum model that satisfies it. Suppose that, for some z such that $z, z + 1 \neq x, y$, it is the case that $(\overline{A}(z), A(z)) = (\overline{A}(z + 1), A(z + 1))$. Consider the model $M' = \langle \mathbb{I}(\mathbb{D}'), V' \rangle$ obtained eliminating the point $z + 1$, along with every interval that starts or ends at $z + 1$. Such an elimination implicitly defines a function $\hat{(\cdot)}$ from points of M to points of M', which is the identical function for every point smaller than z included, and it is the function "-1" for every other point. We can define V', for each propositional letter p, as the union of:

1. $\{[\hat{t}, \hat{t'}] \mid [t, t'] \in V(p) \text{ and } t, t' \neq z\}$,

Algorithm 1 Satisfiability Checker for $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$

```
1: function AĀ□_Horn-Check(φ)
2:     for N ← 2, ..., 6 · |φ| + 2 do
3:         let 𝔻 be a linear domain s.t. |𝔻| = N
4:         for [x, y] ∈ 𝕀(𝔻) do
5:             if Saturate(𝔻, x, y, φ) then return True
6:     return False
```

2. $\{[\hat{t}, \hat{z}] \mid t < z$ and $[t, z], [t, z+1] \in V(p)$ for each $t < z\}$, and
3. $\{[\hat{z}, \hat{t'}] \mid t' > z+1$ and $[z, t'], [z+1, t'] \in V(p)$ for each $t' > z+1\}$.
Unlike Lemma 2, it is straightforward to see that $M', [\hat{x}, \hat{y}] \Vdash \varphi$, as the sets of requests at z are preserved from M to M' It is clear that $|D'| < |D|$; moreover, M' is a minimum model, as its valuation, for each propositional letter, has been obtained by copying or intersecting different components of the valuation of M (which was minimum to begin with). It remains to be shown that, if $N \geq 6 \cdot |\varphi| + 3$, there must exist some z such that $z, z+1 \neq x, y$ and $(\overline{A}(z), A(z)) = (\overline{A}(z+1), A(z+1))$. Let $[A]p_1, [A]p_2, \ldots$ (resp., $[\overline{A}]p_1, [\overline{A}]p_2, \ldots$) be an arbitrary enumeration of A-temporal (resp., $\overline{A}$-temporal) literals, and let us focus our attention to points greater than y. If $(\overline{A}(y+1), A(y+1)) \neq (\overline{A}(y+2), A(y+2))$, then there must be some blocking temporal literal that witnesses such a difference; without loss of generality, let us assume that $[A]p_1$ is a blocking temporal literal. Since M is standard, $[A]p_1 \in A(y+r)$ cannot be blocking for any $r \geq 2$. By iterating this argument, we may conclude that $A(y+|\varphi|) = A(y+|\varphi|+1)$. Now, if $\overline{A}(y+|\varphi|+1) \neq \overline{A}(y+|\varphi|+2)$, we may assume that $[\overline{A}]p_1$ is blocking for $y+|\varphi|+1$. Since M is standard, we can iterate the same argument as before and conclude that $(\overline{A}(y+2\cdot|\varphi|), A(y+2\cdot|\varphi|)) = (\overline{A}(y+2\cdot|\varphi|+1), A(y+2\cdot|\varphi|+1))$. Therefore, in order to guarantee the existence of at least one pair of successive points with precisely the same temporal requests, we need to account for at least $2 \cdot |\varphi|$ points after y, $2\cdot|\varphi|$ points between x and y, $2\cdot|\varphi|$ points before x, plus one, besides x and y themselves. ◀

The above result gives us immediately an NP algorithm for checking the satisfiability of a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$. We show now that the problem is indeed in P by devising a simple deterministic polynomial algorithm. The decision procedure checks all possible minimal models, and for each one of them, all possible starting intervals. For each combination, a saturation procedure iteratively builds a structure $(\mathbb{I}(\mathbb{D}), Hi, Lo)$ that represents a candidate model for the formula, where $\mathbb{D}$ is the underlying domain, Hi labels each interval in $\mathbb{I}(\mathbb{D})$ with the set of formulas 'to be further analyzed' and Lo labels each interval in $\mathbb{I}(\mathbb{D})$ with the set of formulas in $\mathcal{L}(\varphi)$ that holds on it. We first prove that if Saturate terminates with success, then the current structure indeed represents a model for φ on the current starting interval $[x, y]$.

▶ **Lemma 4.** *Let φ be a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, $\mathbb{D}$ a linear domain of length N, and let $[x, y]$ be an interval in $\mathbb{I}(\mathbb{D})$. If Saturate$(\mathbb{D}, x, y, \varphi)$ returns True, then there exists a model M built on $\mathbb{D}$ that satisfies φ on $[x, y]$.*

Proof. Assume that Saturate$(\mathbb{D}, x, y, \varphi)$ returns True, and consider the structure $(\mathbb{I}(\mathbb{D}), Hi, Lo)$ obtained at the end of the procedure. We define a model $M = \langle \mathbb{I}(\mathbb{D}), V \rangle$ such that, for every $p \in \mathcal{AP}$, $V(p) = \{[z, t] \in \mathbb{I}(\mathbb{D}) \mid p \in Lo([z, t])\}$. We first prove that M respects the following property:

$$M, [z, t] \Vdash \psi \text{ for every } [z, t] \in \mathbb{I}(\mathbb{D}) \text{ and } \psi \in Lo([z, t]) \tag{3}$$

Algorithm 2 Saturation

```
1:  function Saturate(𝔻, x, y, φ)
2:      for [z,t] ∈ 𝕀(𝔻) do
3:          Hi([z,t]) ← {(λ1 ∧ ... ∧ λn → λ) | [U](λ1 ∧ ... ∧ λn → λ) ∈ φ}
4:          Lo([z,t]) ← {⊤}
5:      Hi([x,y]) ← Hi([x,y]) ∪ {λ | λ ∈ φ}
6:      while something changes do
7:          let [z,t] ∈ 𝕀(𝔻), ψ ∈ Hi([z,t])
8:          if ψ = p then
9:              Hi([z,t]) ← Hi([z,t]) \ {p}
10:             Lo([z,t]) ← Lo([z,t]) ∪ {p}
11:         else if ψ = [A]p then
12:             Hi([z,t]) ← Hi([z,t]) \ {[A]p}
13:             Lo([z,t]) ← Lo([z,t]) ∪ {[A]p}
14:             for t' > t do
15:                 Lo([t,t']) ← Lo([t,t']) ∪ {p}
16:         else if ψ = [Ā]p then
17:             Hi([z,t]) ← Hi([z,t]) \ {[Ā]p}
18:             Lo([z,t]) ← Lo([z,t]) ∪ {[Ā]p}
19:             for z' < z do
20:                 Lo([z',z]) ← Lo([z',z]) ∪ {p}
21:         else if ψ = λ1 ∧ ... ∧ λn → λ, λ ≠ ⊥ then
22:             if {λ1, ..., λn} ⊆ Lo([z,t]) then
23:                 Hi([z,t]) ← (Hi([z,t]) ∪ {λ}) \ {ψ}
24:                 Lo([z,t]) ← Lo([z,t]) ∪ {ψ}
25:         else if ψ = λ1 ∧ ... ∧ λn → ⊥ then
26:             if {λ1, ..., λn} ⊆ Lo([z,t]) then
27:                 return False
28:         for z ∈ 𝔻 do
29:             for [A]p ∈ ℒ(φ) do
30:                 if ∀t > z(p ∈ Lo([z,t])) then
31:                     for z' < z do
32:                         Lo([z',z]) ← Lo([z',z]) ∪ {[A]p}
33:             for [Ā]p ∈ ℒ(φ) do
34:                 if ∀z' < z(p ∈ Lo([z',z])) then
35:                     for t > z do
36:                         Lo([z,t]) ← Lo([z,t]) ∪ {[Ā]p}
37:     return True
```

We proceed by structural induction on ψ. Consider an interval $[z,t] \in \mathbb{I}(\mathbb{D})$ and a formula $\psi \in Lo([z,t])$:

- if $\psi = \top$ then the property trivially holds;
- if $\psi = p$ for some $p \in \mathcal{AP}$, then the property follows from the definition of M;
- if $\psi = [A]p$ (the case for $[\overline{A}]p$ is symmetrical), we first observe that after the initialization of Saturate (lines 2–5), $Lo([z,t])$ contains only $\top$. Hence, at some point during the execution of Saturate, one of two things may happen: either lines 11–15 move ψ from

$Hi([z,t])$ to $Lo([z,t])$ and add p to $Lo([t,t'])$ for each $t' > t$, or lines 28–36 put ψ in $Lo([z,t])$ because p is in $Lo([t,t'])$ for each $t' > t$. Either way, by inductive hypothesis, we have that $M,[t,t'] \Vdash p$ for every $t' > t$, and thus that $M,[z,t] \Vdash [A]p$;

- if $\psi = \lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda$ and $\lambda \neq \bot$, then at some point during the execution of SATURATE, lines 21–24 move ψ from $Hi([z,t])$ to $Lo([z,t])$ and add λ to $Hi([z,t])$. Some successive iteration of the **while** loop eventually moves λ from $Hi([z,t])$ to $Lo([z,t])$. By inductive hypothesis, we have that $M,[z,t] \Vdash \lambda$ and thus that $M,[z,t] \Vdash \psi$;
- if $\psi = \lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \bot$, then notice that no instruction of SATURATE moves ψ from $Hi([z,t])$ to $Lo([z,t])$. Hence, this case cannot occur.

We can now use (3) to conclude. Suppose by contradiction that $M,[x,y] \nVdash \varphi$: then there exists a formula $\varphi_i \in \varphi$ such that $M,[x,y] \nVdash \psi$. Two cases may arise:

- φ_i is a literal. Then after initialization $\psi \in Hi([x,y])$ (line 5). Since SATURATE eventually moves every literal from Hi to Lo, from (3) we can conclude that $M,[x,y] \Vdash \psi$, a contradiction.
- $\varphi_i = [U](\lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda)$ is a global clause. Since $M,[x,y] \nVdash \psi$, we can find an interval $[z,t]$ such that $M,[z,t] \nVdash \lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda$, that is, $M,[z,t] \Vdash \lambda_1 \wedge \cdots \wedge \lambda_n$ but $M,[z,t] \nVdash \lambda$. We first observe that the initialization of SATURATE (lines 2–5) puts $\lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda$ in $Hi([z,t])$. Now, let λ_i be some literal in the body of the clause; since $M,[z,t] \Vdash \lambda_i$, the following cases arise:
  - if $\lambda_i = p$ then by definition of M we have that $p \in Lo([z,t])$;
  - if $\lambda_i = [A]p$, from $M,[z,t] \Vdash [A]p$ we have that $M,[t,t'] \Vdash p$ for every $t' > t$. By definition of M we have that $p \in Lo([t,t'])$. Hence, at some iteration of the **while** loop, lines 28–36, put $[A]p$ in $Lo([z,t])$;
  - if $\lambda_i = [\overline{A}]p$, we can prove that $[\overline{A}]p \in Lo([z,t])$ as above.

  In all cases $\lambda_i \in Lo([z,t])$ and thus we have that $\{\lambda_1, \ldots, \lambda_n\} \subseteq Lo([z,t])$. Hence, in the case $\lambda \neq \bot$, lines 21–24 eventually move $\lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda$ from $Hi([z,t])$ to $Lo([z,t])$. By (3) we have that $M,[z,t] \Vdash \lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda$, and a contradiction is found. Finally, whenever $\lambda = \bot$, lines 25–27 imply that SATURATE returns False, a contradiction.

Since in all cases we reached a contradiction, we have proved that $M,[x,y] \Vdash \varphi$. ◀

Now we prove that if SATURATE fails then φ has no models of a given length with $[x,y]$ as starting interval.

▶ **Lemma 5.** *Let φ be a set of clauses of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$, $\mathbb{D}$ a linear domain of length N, and let $[x,y]$ be an interval in $\mathbb{I}(\mathbb{D})$. If* SATURATE*$(\mathbb{D},x,y,\varphi)$ returns False, then $M,[x,y] \nVdash \varphi$ for each model M on $\mathbb{D}$.*

Proof. Suppose by contradiction that SATURATE$(\mathbb{D},x,y,\varphi)$ returns False, but there exists a model M on $\mathbb{D}$ that satisfies φ on $[x,y]$. We prove that SATURATE respects the following invariant:

$$\text{For every } [z,t] \in \mathbb{I}(\mathbb{D}) \text{ and } \psi \in Hi([z,t]) \cup Lo([z,t]), \text{ we have that } M,[z,t] \Vdash \psi. \tag{4}$$

After initialization (lines 2–5) we have that for $Hi([z,t])$ contains the scope of every global clause in φ, $Lo([z,t])$ contains $\top$ and that $Hi([x,y])$ contains every initial clause in φ. Hence, (4) follows directly from the fact that $M,[x,y] \Vdash \varphi$.

Now, suppose that (4) holds at the k-th iteration of the **while** loop, and consider the $(k+1)$-th iteration. Let $[z,t]$ and ψ be respectively the interval and the formula selected by line 7. Since $\psi \in Hi([z,t])$ we know by inductive hypothesis that $M,[z,t] \Vdash \psi$. The following cases may arise.

- $\psi = p$. Then lines 8–10 move ψ from $Hi([z,t])$ to $Lo([z,t])$, and (4) is still respected.
- $\psi = [A]p$. Then lines 11–15 move ψ from $Hi([z,t])$ to $Lo([z,t])$, and add p to $Lo([t,t'])$ for every $t' > t$. Since, by inductive hypothesis, $M, [z,t] \Vdash [A]p$, we have that $M, [t,t'] \Vdash p$ and thus (4) is respected.
- $\psi = [\overline{A}]p$. As above.
- $\psi = \lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda$ with $\lambda \neq \bot$. If $\{\lambda_1, \ldots, \lambda_n\} \not\subseteq Lo([z,t])$ then Hi and Lo do not change and thus (4) is respected. If, otherwise, $\{\lambda_1, \ldots, \lambda_n\} \subseteq Lo([z,t])$ then lines 21–24 move ψ from $Hi([z,t])$ to $Lo([z,t])$ and add λ to $Lo([z,t])$. Since for every λ_i we have that $\lambda_i \in Lo([z,t])$, by inductive hypothesis, $M, [z,t] \Vdash \lambda_i$. This implies that $M, [z,t] \Vdash \lambda$ and hence (4) is respected.
- $\psi = \lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \bot$. If $\{\lambda_1, \ldots, \lambda_n\} \not\subseteq Lo([z,t])$ then Hi and Lo do not change and thus (4) is respected. If, otherwise, $\{\lambda_1, \ldots, \lambda_n\} \subseteq Lo([z,t])$ then line 27 terminates the execution of SATURATE by returning False. Since for every λ_i we have that $\lambda_i \in Lo([z,t])$, by inductive hypothesis, $M, [z,t] \Vdash \lambda_i$. Since $M, [z,t] \Vdash \psi$, we have a contradiction, and thus line 27 is never executed.

To conclude, we have to show that (4) is respected also after the execution of lines 28–36. Let $z \in \mathbb{D}$ and $p \in \mathcal{AP}$ be such that $p \in Lo([z,t])$ for every $t > z$. Then, lines 30–32 are eventually executed on z and p, adding $[A]p$ to $Lo([z',z])$ for every $z' < z$. By inductive hypothesis we have that $M, [z,t] \Vdash p$ for every $t > z$ and thus that $M, [z',z] \Vdash [A]p$ for every $z' < z$. This shows that (4) is respected. By the same argument we can also show that for every $z \in \mathbb{D}$ and $p \in \mathcal{AP}$, if $p \in Lo([z',z])$ for every $z' < z$ then the structure Lo is updated in a way that respects the invariant.

Now, observe that we have proved that line 27 cannot be executed, in contradiction with the hypothesis that SATURATE returned False. Hence, M cannot satisfy φ on $[x,y]$. ◀

Lemma 4 and 5 show that SATURATE is correct and complete, and allow us to prove that Algorithm 1 is a deterministic polynomial time solution to the satisfiability problem for $\mathbf{A\overline{A}}^{\Box}_{Horn}$.

▶ **Theorem 6.** *Let φ be a set of clauses of $\mathbf{A\overline{A}}^{\Box}_{Horn}$. Then $\mathbf{A\overline{A}}^{\Box}_{Horn}$-CHECK$(\varphi)$ returns True if and only if φ is satisfiable. Moreover, the time complexity of the procedure is polynomial.*

Proof. Let φ be a satisfiable set of clauses of $\mathbf{A\overline{A}}^{\Box}_{Horn}$. By Lemma 3, we know that there exists an N-minimum standard model M that satisfies φ on some interval $[x,y]$, with $N \leq 6 \cdot |\varphi| + 2$. Since $\mathbf{A\overline{A}}^{\Box}_{Horn}$-CHECK$(\varphi)$ tries to satisfy φ for all model lengths from 2 to $6 \cdot |\varphi| + 2$ and over all intervals $[z,t]$, we have that SATURATE is eventually called on the starting interval $[x,y]$ of a linear domain $\mathbb{D}$ such that $|\mathbb{D}| = N$, returning True (by Lemma 5). Conversely, suppose that $\mathbf{A\overline{A}}^{\Box}_{Horn}$-CHECK$(\varphi)$ returns True. This means that a call to SATURATE$(\mathbb{D}, x, y, \varphi)$ returns True for some linear domain $\mathbb{D}$ and initial interval $[x,y] \in \mathbb{I}(\mathbb{D})$. Then, by Lemma 4, we know that φ is satisfiable.

To prove that the time complexity is polynomial, we start from the complexity of a single call to SATURATE. Every iteration of the **while** loop may move the formula ψ from Hi to Lo and add some new formulas to Lo. Since the number of subformulas of φ is linear in $|\varphi|$ and the number of intervals is quadratic in $|\mathbb{D}|$, after a polynomial number of iterations no new formulas are moved nor added to Lo, and SATURATE terminates. Since the number of intervals in a domain of length N is $\frac{N(N-1)}{2}$, we have that the number of calls to SATURATE in the nested loops of $\mathbf{A\overline{A}}^{\Box}_{Horn}$-CHECK$(\varphi)$ is given by $\sum_{N=2}^{6\cdot|\varphi|+2} \frac{N(N-1)}{2}$, that is a polynomial quantity. Hence, the overall time complexity of $\mathbf{A\overline{A}}^{\Box}_{Horn}$-CHECK is polynomial. ◀

The above theorem shows that the satisfiability problem for $\mathbf{A\overline{A}}^{\Box}_{Horn}$is in P. P-hardness follows form the P-completeness of propositional Horn satisfiability.

▶ **Corollary 7.** *The satisfiability problem for* $\mathbf{A\overline{A}}^{\Box}_{Horn}$ *is* P*-complete.*

4 Intractability

In this section we prove that $\mathbf{A\overline{A}}^{\Box}_{Horn}$ cannot be extended with modalities from the **HS** machinery preserving its computational good behaviour, at least in most cases. We prove that the finite satisfiability $\mathbf{AB}^{\Box}_{Horn}$ becomes hard for PSPACE; then, we extend the result for the fragments obtained by any combination of $[A]$ or $[\overline{A}]$ with any of $[B], [E], [O]$ and their inverses.

To prove that the finite satisfiability problem for $\mathbf{AB}^{\Box}_{Horn}$ is PSPACE-hard, we encode the halting problem of a *Turing machine with polynomial tape* $\mathcal{T} = (Q, \Gamma, \delta, q_0, q_f)$ in it, where Q is the set of states, q_0 is the initial state, q_f is the final accepting state, δ is the transition function, and Γ is the alphabet that contains $0, 1, \sqcup$ (the latter represents the "blank"). Given $n = |\mathcal{T}|$, we assume that $\mathcal{T}$ can use at most $p(n)$ different tape cells, for some polynomial function $p(\cdot)$; under such conditions, the halting problem is PSPACE-hard. Let $\mathcal{T} = (Q, \Gamma, \delta, q_0, q_f)$ be a Turing machine, and let us define (by abusing of the notation) the following set of propositional variables:

$$\mathfrak{L} = \{c_j \mid c \in \Gamma, 0 \leq j \leq p(n)\} \cup \{h_j \mid 0 \leq j \leq p(n)\} \cup Q.$$

The literal $[B]\bot$ uniquely identifies *unit* intervals, that is, intervals of length 1. We use units to encode configurations of $\mathcal{T}$; this is possible since we only have polynomially many different pairs symbol/position, and we can denote each one of them with a different proposition from $\mathfrak{L}$. Moreover, the position of the reading head can be easily encoded in a similar way; for example, h_j denotes that the head is reading the j-th tape symbol. Consider the following formulas to set the initial configurations and the symbols for tape elements, reading head, and current state:

$$\begin{array}{ll}
\phi_1 = q_0 \wedge (\bigwedge_{j=0,\ldots,p(n)} \sqcup_j) \wedge h_0 & \textit{initial configuration} \\
\phi_2 = [U](u \to [B]\bot) \wedge [U]([B]\bot \to u) & \textit{labeling units} \\
\left.\begin{array}{l}
\phi_3 = \bigwedge_{l \in \mathfrak{L}} [U](l \to u) \\
\phi_4 = \bigwedge_{q \neq q', c_j \neq c'_j} [U](q \to \neg q') \wedge [U](c_j \to \neg c'_j) \\
\phi_5 = \bigwedge_{j \neq k} [U](h_j \to \neg h_k)
\end{array}\right\} & \textit{tape/state/head propositions}
\end{array}$$

As far as transitions are concerned, we separate between the actual transitions and the head movement. The transitions are taken care of as follows:

$$\begin{array}{ll}
\left.\begin{array}{l}
\phi_6 = \bigwedge_{c,c',j}^{\delta(q,c)=(q',c',L/R)} [U](((q \wedge c_j \wedge h_j) \to [A]P_{q'}) \wedge ((P_{q'} \wedge u) \to q')) \\
\phi_7 = \bigwedge_{c,c',j}^{\delta(q,c)=(q',c',L/R)} [U](((q \wedge c_j \wedge h_j) \to [A]P_{c'_j}) \wedge ((P_{c'_j} \wedge u) \to c'_j))
\end{array}\right\} & \textit{under head} \\
\phi_8 = \bigwedge_{c,j,k \neq j}^{\delta(q,c)=(q',c',L/R)} [U](((c_k \wedge h_j) \to [A]P_{c_k}) \wedge ((P_{c_k} \wedge u) \to c_k)) & \textit{far from head}
\end{array}$$

The reading head movement is managed as follows:

$$\begin{array}{ll}
\left.\begin{array}{l}
\phi_9 = \bigwedge_{j \neq p(n)}^{\delta(q,c)=(q',c',R)} [U]((h_j \to [A]P_{h_{j+1}}) \wedge ((P_{h_{j+1}} \wedge u) \to h_{j+1})) \\
\phi_{10} = \bigwedge^{\delta(q,c)=(q',c',R)} [U]((h_{p(n)} \to [A]P_{h_{p(n)}}) \wedge ((P_{h_{p(n)}} \wedge u) \to h_{p(n)}))
\end{array}\right\} & \textit{right} \\
\left.\begin{array}{l}
\phi_{11} = \bigwedge_{j \neq 0}^{\delta(q,c)=(q',c',L)} [U]((h_j \to [A]P_{h_{j-1}}) \wedge ((P_{h_{j-1}} \wedge u) \to h_{j-1})) \\
\phi_{12} = \bigwedge^{\delta(q,c)=(q',c',L)} [U]((h_0 \to [A]P_{h_0}) \wedge ((P_{h_0} \wedge u) \to h_0))
\end{array}\right\} & \textit{left}
\end{array}$$

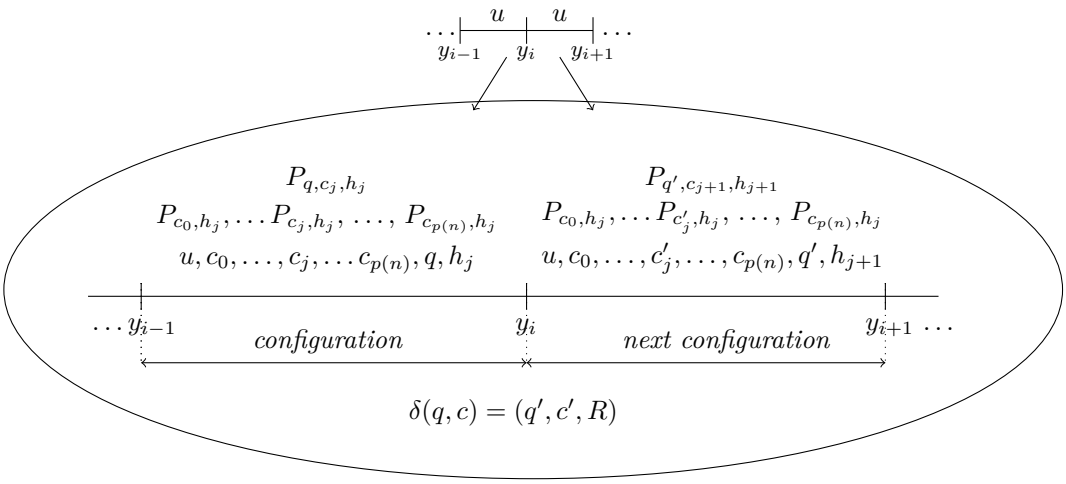


Figure 3 Configurations' structure.

Finally, we make sure that the model is long enough:

$$\phi_{13} = \bigwedge_{q \neq q_f} [U](q \to \neg[A]q)$$

Let $\mathcal{T}$ be a deterministic Turing Machine of size $n = |\mathcal{T}|$, and whose tape is limited by some polynomial function $p(n)$. Then, it is possible to show that $\mathcal{T}$ converges on empty input if and only if the formula:

$$Halts = \phi_1 \wedge \ldots \wedge \phi_{13}$$

is finitely satisfiable. Moreover, since our construction is essentially based on the assumption that the underlying linear order is discrete, it can be easily adapted to obtain an analogous formula for the case of natural numbers, the integers, and the class of all discrete linear ordered sets. An intuitive explanation on the structure of the model obtained in our construction is shown in Fig. 3.

The above construction can be immediately adapted to obtain the same result for any fragment of $\mathbf{HS}^{\Box}_{Horn}$ that includes any combination of $[A]$ or $[\overline{A}]$ with any of $[B], [E]$. For the fragments that include $[O]$ or $[\overline{O}]$, the following considerations are necessary. Consider, for example, the fragment $\mathbf{AO}^{\Box}_{Horn}$. While on infinite discrete linear orders the formula $[O]\bot$ behaves as expected, on finite orders of length N it characterizes unit intervals (as expected) *and* intervals of the type $[x, y_{N-1}]$, where y_{N-1} is the last point of the domain. This does not affect the construction: given a configuration holding on $[x, x+1]$, we may encode the information of the next one in the interval $[x+1, x+2]$ (as desired) as well as in the interval $[x+1, y_{N-1}]$, which in turn, having no A-successors, does not erroneously transmit its information to any other configuration.

▶ **Theorem 8.** *The satisfiability problem for any fragment of* $\mathbf{HS}^{\Box}_{Horn}$ *that features any combination of* $[A]$ *or* $[\overline{A}]$ *with any of* $[B], [E], [O]$ *and their inverses, interpreted in any class of (finite) discrete linearly ordered sets, is* PSPACE*-hard.*

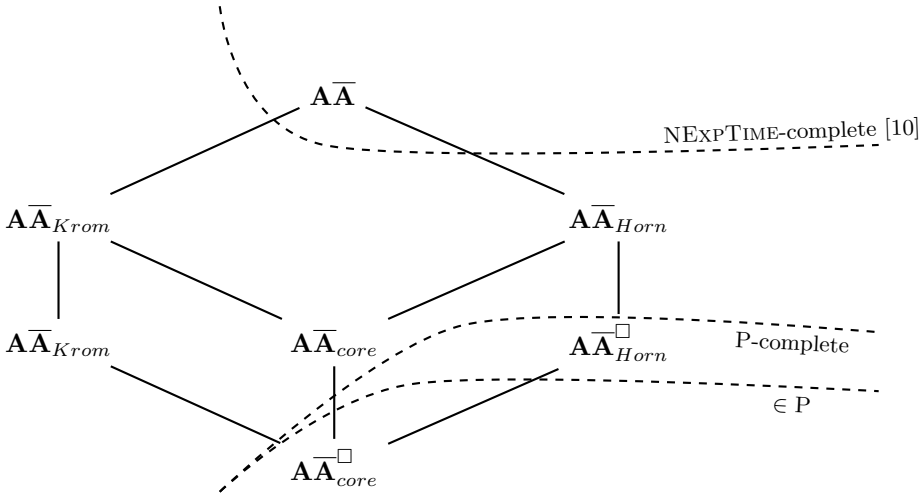


Figure 4 Sub-propositional fragments of $\mathbf{A}\overline{\mathbf{A}}$.

5 Conclusions

In this paper we studied the well-behaved fragments of **HS** known as $\mathbf{A}\overline{\mathbf{A}}$ under the Horn restriction without diamonds ($\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$); we proved that its finite satisfiability problem is P-complete (in sharp constrast with the same problem for $\mathbf{A}\overline{\mathbf{A}}$ without restrictions, which is NExpTime-complete), and that almost every extension of $\mathbf{A}\overline{\mathbf{A}}^{\Box}_{Horn}$ obtained by adding modal operators to it is already intractable. The problems that remain open include the status of $\mathbf{A}\overline{\mathbf{A}}_{Horn}$, as well as extending our results to the discrete infinite case (e.g., $\mathbb{N}$ or $\mathbb{Z}$).

There are very natural motivations for this work; let us hint to some of them. First, Horn propositional logic is the basis for logic programming, which, in turn, is at the core of many Artificial Intelligence applications; although some attempts to extend logical (programming) languages to include time has been done (see, e.g. [3]), further research in this direction is necessary, expecially in the case of interval-based time. Second, in a different context, consider a generic data classification problem, the most common techniques to solve it being decision trees (see, e.g., [21]) and rule extraction algorithms (see, e.g., [19]): from a purely formal point of view, the result of the application of such common algorithms is a set of propositional logic rules, and, although the temporal data classification problem is not nearly as common as its classical counterpart, its development may undoubtedly benefit from studying the syntax, the semantics, and the properties of (interval) temporal rules. Third, it is well known that temporal databases use intervals to represent time [24], and that **HS** is their the natural logical counterpart, which raises the need of tractable fragments of it. Finally, the recent effort of extending description logics with interval temporal capabilities with the introduction of reflexive **HS** [6] raises the natural question of whether similar extensions may be possible with good-behaved fragments of **HS**.

References

1 J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

2 J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

3 D. Anicic, P. Fodor, N. Stojanovic, and R. Stühmer. Computing complex events in an event-driven and logic-based approach. In *Proc. of the 3rd ACM International Conference on Distributed Event-Based Systems (DEBS)*, pages 1–2, 2009.

4 A. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. The complexity of clausal fragments of LTL. In *Proc. of the 19th International Conference Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 35–52, 2013.

5 A. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. A cookbook for temporal conceptual data modelling with description logics. *ACM Transactions on Computational Logic*, 15(3):1–50, 2014.

6 A. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Tractable interval temporal propositional and description logics. In *Proc. of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1417–1423, 2015.

7 B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

8 D. Bresolin, A. Kurucz, E. Muñoz-Velasco, V. Ryzhikov, G. Sciavicco, and M. Zakharyaschev. Horn fragments of the Halpern-Shoham interval temporal logic. *ACM Transactions on Computational Logic*, 2017. in press.

9 D. Bresolin, D. Della Monica, A. Montanari, P. Sala, and G. Sciavicco. Interval temporal logics over strongly discrete linear orders: Expressiveness and complexity. *Theoretical Computer Science*, 560:269–291, 2014.

10 D. Bresolin, A. Montanari, and P. Sala. An optimal tableau-based decision algorithm for Propositional Neighborhood Logic. In *Proc. of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 4393 of *LNCS*, pages 549–560. Springer, 2007.

11 D. Bresolin, E. Muñoz-Velasco, and G. Sciavicco. Sub-propositional fragments of the interval temporal logic of Allen's relations. In *Proc. of the 14th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 8761 of *LNCS*, pages 122–136, 2014.

12 D. Bresolin, E. Muñoz-Velasco, and G. Sciavicco. On the complexity of fragments of horn modal logics. In *Proc. of the 23rd International Symposium on Temporal Representation and Reasoning, (TIME)*, pages 186–195, 2016.

13 D. Bresolin, E. Muñoz-Velasco, and G. Sciavicco. On the expressive power of sub-propositional fragments of modal logic. In *Proc. of the 7th International Symposium on Games, Automata, Logics and Formal Verification (GandALF)*, pages 91–104, 2016.

14 C. C. Chen and I. P. Lin. The computational complexity of satisfiability of temporal Horn formulas in propositional linear-time temporal logic. *Information Processing Letters*, 45(3):131–136, 1993.

15 C. C. Chen and I. P. Lin. The computational complexity of the satisfiability of modal Horn clauses for modal propositional logics. *Theoretical Computer Science Comput*, 129(1):95–121, 1994.

16 L. Fariñas Del Cerro and M. Penttonen. A note on the complexity of the satisfiability of modal Horn clauses. *Journal of Logic Programming*, 4(1):1–10, 1987.

17 J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991.

18 A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16(1):14–21, 1951.

19 S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proc. of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering*, pages 3–24, 2007.

20 M. R. Krom. The decision problem for formulas in prenex conjunctive normal form with binary disjunction. *Journal of Symbolic Logic*, 35(2):14–21, 1970.

21 L. Rokachand O. Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Co., Inc., 2008.

22 L. A. Nguyen. On the complexity of fragments of modal logics. *Advances in Modal Logic*, 5:318–330, 2004.

23 C. H. Papadimitriou. *Computational Complexity*. Wiley, 2003.

24 V. Radhakrishna, P. V. Kumar, and V. Janaki. A survey on temporal databases and data mining. In *Proc. of the International Conference on Engineering (MIS)*, pages 1–6. ACM, 2015.

Improved Set-Based Symbolic Algorithms for Parity Games*†

Krishnendu Chatterjee[1], Wolfgang Dvořák[2], Monika Henzinger[3], and Veronika Loitzenbauer[4]

1 **IST, Klosterneuburg, Austria**
2 **TU Wien, Vienna, Austria; and University of Vienna, Vienna, Austria**
3 **University of Vienna, Vienna, Austria**
4 **Bar-Ilan University, Ramat Gan, Israel; and University of Vienna, Vienna, Austria**

Abstract

Graph games with ω-regular winning conditions provide a mathematical framework to analyze a wide range of problems in the analysis of reactive systems and programs (such as the synthesis of reactive systems, program repair, and the verification of branching time properties). Parity conditions are canonical forms to specify ω-regular winning conditions. Graph games with parity conditions are equivalent to μ-calculus model checking, and thus a very important algorithmic problem. Symbolic algorithms are of great significance because they provide scalable algorithms for the analysis of large finite-state systems, as well as algorithms for the analysis of infinite-state systems with finite quotient. A set-based symbolic algorithm uses the basic set operations and the one-step predecessor operators. We consider graph games with n vertices and parity conditions with c priorities (equivalently, a μ-calculus formula with c alternations of least and greatest fixed points). While many explicit algorithms exist for graph games with parity conditions, for set-based symbolic algorithms there are only two algorithms (notice that we use space to refer to the number of sets stored by a symbolic algorithm):

(a) the basic algorithm that requires $O(n^c)$ symbolic operations and linear space; and (b) an improved algorithm that requires $O(n^{c/2+1})$ symbolic operations but also $O(n^{c/2+1})$ space (i.e., exponential space). In this work we present two set-based symbolic algorithms for parity games:

(b) our first algorithm requires $O(n^{c/2+1})$ symbolic operations and only requires linear space; and (b) developing on our first algorithm, we present an algorithm that requires $O(n^{c/3+1})$ symbolic operations and only linear space.

We also present the first linear space set-based symbolic algorithm for parity games that requires at most a sub-exponential number of symbolic operations.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases model checking, graph games, parity games, symbolic computation, progress measure

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.18

* All authors are partially supported by the Vienna Science and Technology Fund (WWTF) through project ICT15-003. K. C. is partially supported by the Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE) and an ERC Start grant (279307: Graph Games). W. D., M. H., and V. L. received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506. V. L. is partially supported by the ISF grant #1278/16 and an ERC Consolidator Grant (project MPM).

† Some proofs are omitted due to space restrictions, but a full version with all proofs is available at [12].

1 Introduction

In this work we present improved set-based symbolic algorithms for solving graph games with parity winning conditions, which is equivalent to modal μ-calculus model-checking.

Graph games. Two-player graph games provide the mathematical framework to analyze several important problems in computer science, especially in formal methods for the analysis of reactive systems. Graph games are games that proceed for an infinite number of rounds, where the two players take turns to move a token along the edges of the graph to form an infinite sequence of vertices (which is called a *play* or a *trace*). The desired set of plays is described as an ω-regular winning condition. A *strategy* for a player is a recipe that describes how the player chooses to move tokens to extend plays, and a *winning* strategy ensures the desired set of plays against all strategies of the opponent. Some classical examples of graph games in formal methods are as follows:

(a) If the vertices and edges of a graph represent the states and transitions of a reactive system, resp., then the synthesis problem (Church's problem [14]) asks for the construction of a *winning strategy* in a graph game [8, 36, 35, 33, 34].

(b) The problems of
  - **(i)** verification of a branching-time property of a reactive system [19], where one player models the existential quantifiers and the opponent models the universal quantifiers; as well as
  - **(ii)** verification of open systems [2], where one player represents the controller and the opponent represents the environment;

  are naturally modeled as graph games, where the winning strategies represent the choices of the existential player and the controller, respectively.

Moreover, game-theoretic formulations have been used for refinement [25], compatibility checking [17] of reactive systems, program repair [28], and synthesis of programs [11]. Graph games with *parity winning* conditions are particularly important since all ω-regular winning conditions (such as safety, reachability, liveness, fairness) as well as all Linear-time Temporal Logic (LTL) winning conditions can be translated to parity conditions [37, 38], and parity games are equivalent to modal μ-calculus model checking [19]. In a parity winning condition, every vertex is assigned a non-negative integer priority from $\{0, 1, \ldots, c-1\}$, and a play is winning if the highest priority visited infinitely often is even. Graph games with parity conditions can model all the applications mentioned above, and there is a rich literature on the algorithmic study of finite-state parity games [19, 6, 40, 29, 43, 31, 39].

Explicit vs. symbolic algorithms. The algorithms for parity games can be classified broadly as *explicit* algorithms, where the algorithms operate on the explicit representation of the graph game, and *implicit or symbolic* algorithms, where the algorithms only use a set of predefined operations and do not explicitly access the graph game. Symbolic algorithms are of great significance for the following reasons:

(a) first, symbolic algorithms are required for large finite-state systems that can be succinctly represented implicitly (e.g., programs with Boolean variables) and symbolic algorithms are scalable, whereas explicit algorithms do not scale; and

(b) second, for infinite-state systems (e.g., real-time systems modeled as timed automata, or hybrid systems, or programs with integer domains) only symbolic algorithms are applicable, rather than explicit algorithms. Hence for the analysis of large systems or infinite-state systems symbolic algorithms are necessary.

Significance of set-based symbolic algorithms. The most significant class of symbolic algorithms for parity games are based on *set operations*, where the allowed symbolic operations are:

(a) basic set operations such as union, intersection, complement, and inclusion; and
(b) one step predecessor (Pre) operations.

Note that the basic set operations (that only involve state variables) are much cheaper as compared to the predecessor operations (that involve both variables of the current and of the next state). Thus in our analysis we will distinguish between the basic set operations and the predecessor operations. We refer to the number of sets stored by a set-based symbolic algorithm as its space. The significance of set-based symbolic algorithms is as follows:

(a) First, in several domains of the analysis of both infinite-state systems (e.g., games over timed automata or hybrid systems) as well as large finite-state systems (e.g., programs with many Boolean variables, or bounded integer variables), the desired model-checking question is specified as a μ-calculus formula with the above set operations [18, 16]. Thus an algorithm with the above set operations provides a symbolic algorithm that is directly applicable to the formal analysis of such systems.
(b) Second, in other domains such as in program analysis, the one-step predecessor operators are routinely used (namely, with the weakest-precondition as a predicate transformer). A symbolic algorithm based only on the above operations thus can easily be developed on top of the existing implementations. Moreover, recent work [4] shows how efficient procedures (such as constraint-based approaches using SMTs) can be used for the computation of the above operations in infinite-state games. This highlights that symbolic one-step operations can be applied to a large class of problems.
(c) Finally, if a symbolic algorithm is described with the above very basic set of operations, then any practical improvement to these operations in a particular domain would translate to a symbolic algorithm that is faster in practice for the respective domain.

Thus the problem is practically relevant, and understanding the symbolic complexity of parity games is an interesting and important problem.

Previous results. We summarize the main previous results for finite-state game graphs with parity conditions. Consider a parity game with n vertices, m edges, and c priorities (which is equivalent to μ-calculus model-checking of transitions systems with n states, m transitions, and a μ-calculus formula of alternation depth c). In the interest of concise presentation, in the following discussion, we ignore denominators in c in the running time bounds, see Theorems 7 and 8, and the references for precise bounds.

Let us first consider *set-based symbolic algorithms*. Recall that we use space to refer to the number of sets stored by a symbolic algorithm. The basic set-based symbolic algorithm (based on the direct evaluation of the nested fixed point of the μ-calculus formula) for parity games requires $O(n^c)$ symbolic operations and space linear in c [20]. In a breakthrough result [6], a new set-based symbolic algorithm was presented that requires $O(n^{c/2+1})$ symbolic operations, but also requires $O(n^{c/2+1})$ many sets, i.e., exponential space as compared to the linear space of the basic algorithm. A simplification of the result of [6] was presented in [40].

Now consider *explicit algorithms* for parity games. The classical algorithm requires $O(n^{c-1}m)$ time and can be implemented in quasi-linear space [44, 34], which was then improved to the small-progress measure algorithm that requires $O(n^{c/2}m)$ time and space to store $O(c \cdot n)$ integer counters [29]. The small-progress measure algorithm, which is an explicit algorithm, uses an involved domain of the product of integer priorities and *lift* operations (which is a lexicographic max and min in the involved domain). The algorithm shows that the fixed point of the lift operation computes the solution of the parity game.

The lift operation can be encoded with algebraic binary decision diagrams [9] but this does not provide a set-based symbolic algorithm. Other notable explicit algorithms for parity games are as follows:

(a) a strategy improvement algorithm [43], which in the worst-case is exponential [22];

(b) a dominion-based algorithm [31] that requires $n^{O(\sqrt{n})}$ time and a randomized $n^{O(\sqrt{n/\log n})}$ algorithm [5] (both algorithms are sub-exponential, but inherently explicit algorithms); and, combining the small-progress measure and the dominion-based algorithm,

(c) an $O(n^{c/3}m)$ time algorithm [39] and its improvement for dense graphs with c sub-polynomial in n to an $O(n^{c/3}n^{4/3})$ time algorithm [13] (both bounds are simplified).

A recent breakthrough result [10] shows that parity games can be solved in $O(n^{\log c})$ time, i.e., quasi-polynomial time. Follow-up work [30, 21] reduced the space requirements from quasi-polynomial to $O(n \log n \log c)$, i.e., to quasi-linear, space.

While the above algorithms are specified for finite-state graphs, the symbolic algorithms also apply to infinite-state graphs with a finite bi-simulation quotient (such as timed-games, or rectangular hybrid games), and then n represents the size of the finite quotient.

Our contributions. Our results for game graphs with n vertices and parity objectives with c priorities are as follows.

1. First, we present a set-based symbolic algorithm that requires $O(n^{c/2+1})$ symbolic operations and linear space (i.e., a linear number of sets). Thus it matches the symbolic operations bound of [6, 40] and brings the space requirements down to a linear number of sets as in the classical algorithm (albeit linear in n and not in c).
2. Second, developing on our first algorithm, we present a set-based symbolic algorithm that requires $O(n^{c/3+1})$ symbolic operations (simplified bound) and linear space. Thus it improves the symbolic operations of [6, 40] while achieving an exponential improvement in the space requirement. We also present a modification of our algorithm that requires $n^{O(\sqrt{n})}$ symbolic operations and at most linear space. This is the first linear-space set-based symbolic algorithm that requires at most a sub-exponential number of symbolic operations.

In the results above the number of symbolic operations mentioned is the number of predecessor operations, and in all cases the number of required basic set operations (which are usually cheaper) is at most a factor of $O(n)$ more. Our main results and comparison with previous set-based symbolic algorithms are presented in the table below.

reference	symbolic operations	space
[20, 44]	$O(n^c)$	$O(c)$
[6, 40]	$O(n^{c/2+1})$	$O(n^{c/2+1})$
Thm. 7	$\mathbf{O(n^{c/2+1})}$	$\mathbf{O(n)}$
Thm. 8	$\min\{\mathbf{n^{O(\sqrt{n})}}, \mathbf{O(n^{c/3+1})}\}$	$\mathbf{O(n)}$

Our *technical contributions* are as follows. We provide a symbolic version of the progress measure algorithm. The main challenge is to succinctly encode the numerical domain of the progress measure as sets. More precisely, the challenge is to represent $\Theta(n^{c/2})$ many numerical values with $O(n)$ many sets, such that they can still be efficiently processed by a set-based symbolic algorithm. For the sake of efficiency our algorithms consider sets S_r storing all vertices with progress measure at least r. However, there are $\Theta(n^{c/2})$ many such sets S_r and thus, to reduce the space requirements to a linear number of sets, we use a succinct representation that encodes all the sets S_r with just $O(n)$ many sets, such that we can restore a set S_r efficiently whenever it is processed by the algorithm.

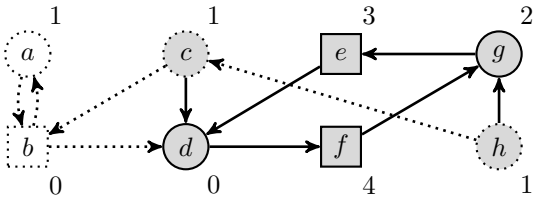


Figure 1 A parity game with 5 priorities. Circles denote player $\mathcal{E}$ vertices, squares denote player $\mathcal{O}$ vertices. The numeric label of a vertex gives its priority, e.g., a is an $\mathcal{E}$-vertex with priority 1. The set of solid vertices, i.e., the set $\{d, e, f, g\}$, is a player-$\mathcal{E}$ dominion and the union of this $\mathcal{E}$-dominion with the vertices c and h is the winning set of player $\mathcal{E}$ in this game. The solid edges indicate a winning strategy for player $\mathcal{E}$.

2 Preliminaries and Previous Results

2.1 Parity Games

We consider *games on graphs* played by two adversarial players, denoted by $\mathcal{E}$ (for even) and $\mathcal{O}$ (for odd). We use z to denote one of the players of $\{\mathcal{E}, \mathcal{O}\}$ and $\overline{z}$ to denote its opponent. We denote by (V, E) a directed graph with $n = |V|$ vertices and $m = |E|$ edges, where V is the vertex set and E is the edge set. A *game graph* $\mathcal{G} = ((V, E), (V_{\mathcal{E}}, V_{\mathcal{O}}))$ is a directed graph (V, E) with a partition of the vertices into player-$\mathcal{E}$ vertices $V_{\mathcal{E}}$ and player-$\mathcal{O}$ vertices $V_{\mathcal{O}}$. For a vertex $u \in V$, we write $\mathit{Out}(u) = \{v \in V \mid (u, v) \in E\}$ for the set of successor vertices of u. As a standard convention (for technical simplicity) we consider that every vertex has at least one outgoing edge, i.e., $\mathit{Out}(u)$ is non-empty for all vertices u.

A game is initialized by placing a token on a vertex. Then the two players form an infinite path, called *play*, in the game graph by moving the token along the edges. Whenever the token is on a vertex of V_z, player z moves the token along one of the outgoing edges of the vertex. Formally, a *play* is an infinite sequence $\langle v_0, v_1, v_2, \ldots \rangle$ of vertices such that $(v_j, v_{j+1}) \in E$ for all $j \geq 0$.

A *parity game* $\mathcal{P} = (\mathcal{G}, \alpha)$ with c priorities consists of a game graph $G = ((V, E), (V_{\mathcal{E}}, V_{\mathcal{O}}))$ and a *priority function* $\alpha : V \to [c]$ that assigns an integer from the set $[c] = \{0, \ldots, c-1\}$ to each vertex (see Figure 1 for an example). Player $\mathcal{E}$ (resp. player $\mathcal{O}$) wins a play of the parity game if the *highest* priority occurring infinitely often in the play is even (resp. odd). We denote by P_i the set of vertices with priority i, i.e., $P_i = \{v \in V \mid \alpha(v) = i\}$. Note that if P_i is empty for $0 < i < c-1$, then the priorities $> i$ can be decreased by 2 without changing the parity condition, and when P_{c-1} is empty, we simply have a parity game with a priority less; thus we assume w.l.o.g. $P_i \neq \emptyset$ for $0 < i < c$.

A *strategy* of a player $z \in \{\mathcal{E}, \mathcal{O}\}$ is a function that, given a finite prefix of a play ending at $v \in V_z$, selects a vertex from $\mathit{Out}(v)$ to extend the finite prefix. *Memoryless strategies* depend only on the last vertex of the finite prefix. That is, a memoryless strategy of player z is a function $\sigma : V_z \to V$ such that for all $v \in V_z$ we have $\sigma(v) \in \mathit{Out}(v)$. It is well-known that for parity games it is sufficient to consider memoryless strategies [19, 34]. Therefore we only consider memoryless strategies from now on. A start vertex v, a strategy σ for $\mathcal{E}$, and a strategy π for $\mathcal{O}$ describe a unique play $\omega(v, \sigma, \pi) = \langle v_0, v_1, v_2, \ldots \rangle$, which is defined as follows: $v_0 = v$ and for all $i \geq 0$, if $v_i \in V_{\mathcal{E}}$, then $\sigma(v_i) = v_{i+1}$, and if $v_i \in V_{\mathcal{O}}$, then $\pi(v_i) = v_{i+1}$.

A strategy σ is *winning* for player $\mathcal{E}$ at start vertex v iff for all strategies π of player $\mathcal{O}$ we have that the play $\omega(v, \sigma, \pi)$ satisfies the parity condition, and analogously for winning strategies for player $\mathcal{O}$. A vertex v belongs to the *winning set* W_z of player z if player z has

a winning strategy from start vertex v. Every vertex is winning for exactly one of the two players. The algorithmic problem we study for parity games is to compute the winning sets of the two players. A non-empty set of vertices D is a *player-z dominion* if player z has a winning strategy from every vertex of D that also ensures only vertices of D are visited.

2.2 Set-based Symbolic Operations

Symbolic algorithms operate on sets of vertices, which are usually described by Binary Decision Diagrams (BDD) [32, 1]. For the symbolic algorithms for parity games we consider the most basic form of symbolic operations, namely, *set-based symbolic* operations. More precisely, we only allow the following operations:

Basic set operations. First, we allow basic set operations like $\cup$, $\cap$, $\setminus$, $\subseteq$, and $=$.

One-step operations. Second, we allow the following symbolic one-step operations:

(a) the one-step predecessor operator $\mathsf{Pre}(B) = \{v \in V \mid \exists u \in B : (v,u) \in E\}$; and

(b) the one-step *controllable* predecessor operator $\mathsf{CPre}_z(B) = \{v \in V_z \mid \mathit{Out}(v) \cap B \neq \emptyset\} \cup \{v \in V_{\overline{z}} \mid \mathit{Out}(v) \subseteq B\}$; i.e., the CPre_z operator computes all vertices from which z can ensure that in the next step the successor belongs to the given set B. Moreover, the CPre_z operator can be defined using the Pre operator and basic set operations as follows: $\mathsf{CPre}_z(B) = \mathsf{Pre}(B) \setminus (V_{\overline{z}} \cap \mathsf{Pre}(V \setminus B))$.

Algorithms that use only the above operations are called *set-based symbolic* algorithms. Additionally, successor operations can be allowed but are not needed for our algorithms. The above symbolic operations correspond to primitive operations in standard symbolic packages like CuDD [41].

Typically, the basic set operations are cheaper (as they encode relationships between state variables) as compared to the one-step symbolic operations (which encode the transitions and thus the relationship between the variables of the present and of the next state). Thus in our analysis we distinguish between these two types of operations.

For the *space* requirements of set-based symbolic algorithms, as per standard convention [6, 9], we consider that a set is stored in constant space (e.g., a set can be represented symbolically as one BDD [7]). We thus consider the space requirement of a symbolic algorithm to be the maximal number of sets that the algorithm has to store.

2.3 Progress Measure Algorithm

We first provide basic intuition for the progress measure [29] and then provide the formal definitions. Solving parity games can be reduced to computing the progress measure [29]. In Section 3 we present a set-based symbolic algorithm to compute the progress measure.

High-level intuition. Towards a high-level intuition behind the progress measure, consider an $\mathcal{E}$-dominion D, i.e., player $\mathcal{E}$ wins on all vertices of D without leaving D. Fix a play started at a vertex $u \in D$ in which player $\mathcal{E}$ follows her winning strategy on D. In the play from some point on the highest priority visited by the play, say α^*, has to be even. Let v_* be the vertex after which the highest visited priority is α^* (recall that memoryless strategies are sufficient for parity games). Before v_* is visited, the play might have visited vertices with odd priority higher than α^* but the number of these vertices has to be less than n. The *progress measure* is based on a so-called *lexicographic ranking* function that assigns a *rank* to each vertex v, where the rank is a "vector of counters" for the number of times player $\mathcal{O}$ can force a play to visit an odd priority vertex before a vertex with higher even priority is reached. If player $\mathcal{O}$ can ensure a counter value of at least n, then she can ensure that a

cycle with highest priority odd is reached from v and therefore player $\mathcal{E}$ cannot win from the vertex v. Conversely, if player $\mathcal{O}$ can reach a cycle with highest priority odd before reaching a higher even priority, then she can also force a play to visit an odd priority n times (thus a counter value of n) before reaching a higher even priority. In other words, a vertex u is in the $\mathcal{E}$-dominion D if and only if player $\mathcal{O}$ *cannot* force any counter value to reach n from u. When a vertex u is classified as winning for player $\mathcal{O}$, it is marked with the rank $\top$ and whenever $\mathcal{O}$ has a strategy for some vertex v to reach a $\top$-ranked vertex, it is also winning for player $\mathcal{O}$ and thus ranked $\top$. Computing the progress measure is done by updating the rank of a vertex according to the ranks of its successors and is equal to computing the least simultaneous fixed point for all vertices with respect to "ranking functions".

An additional property of the progress measure is that the ranks assigned to the vertices of the $\mathcal{E}$-dominion provide a certificate for a winning strategy of player $\mathcal{E}$ within the dominion, namely, player $\mathcal{E}$ can follow edges that lead to vertices with "lower or equal" rank with respect to a specific ordering of the ranks.

Formal definitions. We next provide formal definitions of *rank*, the *ranking function*, the ordering on the ranks, the *lift*-operators, and finally the *progress measure* (see also [29]).

We start with *the progress measure domain* M_G^∞ and consider parity games with n vertices and priorities $[c]$. Let n_i be the number of vertices with priority i for odd i (i.e., $n_i = |P_i|$), let $n_i = 0$ for even i, and let $N_i = [n_i + 1]$ for $0 \leq i < c$. Let $M_{\mathcal{G}} = (N_0 \times N_1 \times \cdots \times N_{c-2} \times N_{c-1})$ be the product domain where every even index is 0 and every odd index i is a number between 0 and n_i. The *progress measure domain* is $M_{\mathcal{G}}^\infty = M_{\mathcal{G}} \cup \{\top\}$, where $\top$ is a special element called the top element. Then we have $|M_{\mathcal{G}}^\infty| = 1 + \prod_{i=1}^{\lfloor c/2 \rfloor}(n_{2i-1}+1) = O\big(\big(\frac{n}{\lfloor c/2 \rfloor}\big)^{\lfloor c/2 \rfloor}\big)$ [29] (this bound uses that w.l.o.g. $|P_i| > 0$ for each priority $i > 0$).

A *ranking function* $\rho : V \to M_{\mathcal{G}}^\infty$ assigns to each vertex a *rank* r that is either one of the c dimensional vectors in $M_{\mathcal{G}}$ or the top element $\top$. Note that a rank has at most $\lfloor c/2 \rfloor$ non-zero entries. Informally, we call the entries of a rank with an odd index i a "counter" because as long as the top element is not reached, it counts (with "carry", i.e., if n_i is reached, the next highest counter is increased by one and the counter at index i is reset to zero) the number of times a vertex of priority i is reached before a vertex of higher priority is reached (from some specific start vertex). The co-domain of ρ is $M_{\mathcal{G}}^\infty = M_{\mathcal{G}} \cup \{\top\}$ and we index the elements of the vectors from 0 to $c-1$.

We use the *lexicographic comparison operator* $<$ of the ranks assigned by ρ: the vectors are considered in the lexicographical order, where the left most entry is the least significant one and the right most entry is the most significant one, and $\top$ is the maximum element of the ordering. We write $\bar{0}$ to refer to the all zero vector (i.e., the minimal element of the ordering) and $\bar{N}$ to refer to the maximal vector $(n_0, n_1, \ldots, n_{c-1})$ (i.e., the second largest element, after $\top$, in the ordering).

Next we introduce the *lexicographic increment and decrement operations.* Given a rank r, i.e., either a vector or $\top$, we refer to the successor in the ordering $<$ by $\mathrm{inc}(r)$ (with $\mathrm{inc}(\top) = \top$), and to the predecessor in the ordering $<$ by $\mathrm{dec}(r)$ (with $\mathrm{dec}(\bar{0}) = \bar{0}$). We also consider restrictions of inc and dec to fewer dimensions, which are described below. Given a vector $x = (x_0, x_1, x_2, \ldots, x_{c-1})$, we denote by $\langle x \rangle_\ell$ (for $0 \leq \ell < c$) the vector $(0, 0, \ldots, 0, x_\ell, \ldots, x_{c-1})$, where we set all elements with index less than ℓ to 0; in particular $x = \langle x \rangle_0$. Intuitively, we use the notation $\langle x \rangle_\ell$ to "reset the counters" for priorities lower than ℓ when a vertex of priority ℓ is reached (as long as we have not counted up to the top element). Moreover, we also generalize the ordering to a family of orderings $<_\ell$ where $x <_\ell y$ for two vectors x and y iff $\langle x \rangle_\ell < \langle y \rangle_\ell$; the top element $\top$ is the maximum element of each

ordering. In particular, $x <_0 y$ iff $x < y$ and in our setting also $x <_1 y$ iff $x < y$. We further have restricted versions inc_ℓ and dec_ℓ of inc and dec; note that dec_ℓ is a partial function and that ℓ will be the priority of the vertex v for which we want to update its rank and x will be the rank of one of its neighbors in the game graph.

- $\mathrm{inc}_\ell(x)$: For $x = \top$ we have $\mathrm{inc}_\ell(\top) = \top$; Otherwise $\mathrm{inc}_\ell(x) = \langle x \rangle_\ell$ if ℓ is even and $\mathrm{inc}_\ell(x) = \min\{y \in M_\mathcal{G}^\infty \mid y >_\ell x\}$ if ℓ is odd.
- $\mathrm{dec}_\ell(x) : \mathrm{dec}_\ell(x) = \bar{0}$ if $\langle x \rangle_\ell = \bar{0}$; Otherwise if $\langle x \rangle_\ell > \bar{0}$ then $\mathrm{dec}_\ell(x) = \min\{y \in M_\mathcal{G} \mid x = \mathrm{inc}_\ell(y)\}$.

For $\bar{0} < \langle x \rangle_\ell < \top$ we have $\mathrm{inc}_\ell(\mathrm{dec}_\ell(x)) = \mathrm{dec}_\ell(\mathrm{inc}_\ell(x)) = \langle x \rangle_\ell$ while for $\top$ we only have $\mathrm{inc}_\ell(\mathrm{dec}_\ell(\top)) = \top$ and for $\langle x \rangle_\ell = \bar{0}$ only $\mathrm{dec}_\ell(\mathrm{inc}_\ell(x)) = \bar{0}$. By the restriction of inc by the priority ℓ of v, for both even and odd priorities the counters for lower (odd) priorities are reset to zero as long as the top element is not reached. For an odd ℓ additionally the counter for ℓ is increased or, if the counter for ℓ has already been at n_ℓ, then one of the higher counters is increased while the counter for ℓ is reset to zero as well; if no higher counter can be increased any more, then the rank of v is set to $\top$.

Recall the interpretation of the progress measure as a witness for a player-$\mathcal{E}$ winning strategy on an $\mathcal{E}$-dominion, where player $\mathcal{E}$ wants to follow a path of non-increasing rank. The function best we define next reflects the ability of player $\mathcal{E}$ to choose the edge leading to the lowest rank when he owns the vertex, while for player-$\mathcal{O}$ vertices all edges need to lead to non-increasing ranks if player $\mathcal{E}$ can win from this vertex. The function best for each vertex v and ranking function ρ is given by

$$\mathrm{best}(\rho, v) = \begin{cases} \min\{\rho(w) \mid (v,w) \in E\} & \text{if } v \in V_\mathcal{E}\,, \\ \max\{\rho(w) \mid (v,w) \in E\} & \text{if } v \in V_\mathcal{O}\,. \end{cases}$$

Finally, the lift operation implements the incrementing of the rank of a vertex v according to its priority and the ranks of its neighbors:

$$\mathrm{Lift}(\rho, v)(u) = \begin{cases} \mathrm{inc}_{\alpha(v)}(\mathrm{best}(\rho, v)) & \text{if } u = v\,, \\ \rho(u) & \text{otherwise}\,. \end{cases}$$

The $\mathrm{Lift}(., v)$-operators are monotone and the *progress measure* for a parity game is defined as the *least simultaneous fixed point of all* $\mathrm{Lift}(., v)$*-operators.* The progress measure can be computed by starting with the ranking function equal to the all-zero function and iteratively applying the $\mathrm{Lift}(., v)$-operators in an arbitrary order [29]. Note that in this case the $\mathrm{Lift}(., v)$-operator assigns only rank vectors r with $r = \langle r \rangle_{\alpha(v)}$ to v. See [29] for a worst-case example for any lifting algorithm. By [29], the winning set of player $\mathcal{E}$ can be obtained from the progress measure by selecting those vertices whose rank is a vector, i.e., smaller than $\top$.

▶ **Lemma 1** ([29]). *For a given parity game and the progress measure ρ with co-domain $M_\mathcal{G}^\infty$, the set of vertices with $\rho(v) < \top$ is exactly the winning set of player $\mathcal{E}$.*

This implies that to solve parity games it is sufficient to provide an algorithm that computes the least simultaneous fixed point of all $\mathrm{Lift}(., v)$-operators. The Lift operation can be computed explicitly in $O(m)$ time, which gives the SmallProgressMeasure algorithm of [29]. The SmallProgressMeasure algorithm is an explicit algorithm that requires $O(m \cdot |M_\mathcal{G}^\infty|) = O\big(m \cdot \big(\frac{n}{\lfloor c/2 \rfloor}\big)^{\lfloor c/2 \rfloor}\big)$ time and $O(n \cdot c)$ space (assuming constant size integers).

3 Set-based Symbolic Progress Measure Algorithm for Parity Games

In this section we present a set-based symbolic algorithm for parity games, with n vertices and c priorities, by showing how to compute a progress measure (see Section 2.3) using only set-based symbolic operations (see Section 2.2). All proofs are provided in Appendix A. We mention the *key differences* of Algorithm SymbolicParityDominion and the explicit progress-measure algorithm ([29], see Section 2.3).

1. The main challenge for an efficient set-based symbolic algorithm similar to the SmallProgressMeasure algorithm is to represent $\Theta(n^{c/2})$ many numerical values succinctly with $O(n)$ many sets, such that they can still be efficiently processed by a symbolic algorithm.
2. To exploit the power of symbolic operations, in each iteration of the algorithm we compute all vertices whose rank can be increased to a certain value r. This is in sharp contrast to the explicit progress-measure algorithm, where vertices are considered one by one and the rank is increased to the maximal possible value.

Key concepts. Recall that the progress measure for parity games is defined as the least simultaneous fixed point of the Lift(ρ, v)-operators on a ranking function $\rho : V \to M_{\mathcal{G}}^{\infty}$. There are two key aspects of our algorithm:

1. *Symbolic encoding of numerical domain.* In our symbolic algorithm we cannot directly deal with the ranking function but have to use sets of vertices to encode it. We first formulate our algorithm with sets S_r for $r \in M_{\mathcal{G}}^{\infty}$ that contain all vertices that have rank r or higher; that is, given a function ρ, the corresponding sets are $S_r = \{v \mid \rho(v) \geq r\}$. On the other hand, given a family of sets $\{S_r\}_r$, the corresponding ranking function $\rho_{\{S_r\}_r}$ is given by $\rho_{\{S_r\}_r}(v) = \max\{r \in M_{\mathcal{G}}^{\infty} \mid v \in S_r\}$. This formulation encodes the numerical domain with sets but uses exponential in c many sets.
2. *Space efficiency.* We refine the algorithm to directly encode the ranks with one set for each possible index-value pair. This reduces the required number of sets to linear at the cost of increasing the number of set operations only by a factor of n; the number of one-step symbolic operations does not increase.

We first present the variant that uses an exponential number of sets and then show how to reduce the number of sets to linear.

The above ideas yield a set-based symbolic algorithm, but since we now deal with sets of vertices, as compared to individual vertices, the correctness needs to be established. The non-trivial aspect of the proof is to identify appropriate *invariants on sets* (which we call *symbolic invariants*, see Invariant 3) and use them to establish the correctness.

3.1 The Set-based Symbolic Progress Measure Algorithm

The codomain M_h^{∞}. We formulate our algorithm such that it cannot only compute the winning sets of the players but also $\mathcal{E}$-dominions of size at most $h+1$. (For $\mathcal{O}$-dominions add one to each priority and exchange the roles of the two players.) The only change needed for this is to use the codomain M_h^{∞}, instead of $M_{\mathcal{G}}^{\infty}$, for the inc and dec operations. The codomain M_h^{∞} contains all ranks of $M_{\mathcal{G}}^{\infty}$ whose entries sum up to at most h.

The sets S_r and the ranking function $\rho_{\{S_r\}_r}$. The algorithm implicitly maintains a rank for each vertex. A vertex is contained in a set S_r only if its maintained rank is *at least* r. Each set S_r is monotonically increasing throughout the algorithm. The rank of a vertex v is the highest r such that $v \in S_r$. In other words, the family of sets $\{S_r\}_r$ defines the ranking function $\rho_{\{S_r\}_r}(v) = \max\{r \in M_h^{\infty} \mid v \in S_r\}$. When the rank of a vertex is increased,

Algorithm SymbolicParityDominion: Symbolic Progress Measure Algorithm

Input : *parity game* $\mathcal{P} = (\mathcal{G}, \alpha)$, with *game graph* $\mathcal{G} = ((V, E), (V_\mathcal{E}, V_\mathcal{O}))$, *priority function* $\alpha : V \to [c]$, and *parameter* $h \in [0, n] \cap \mathbb{N}$
Output : Set containing all $\mathcal{E}$-dominions of size $\leq h + 1$, which is an $\mathcal{E}$-dominion or empty.

```
1  S_0̄ ← V; S_r ← ∅ for r ∈ M_h^∞ \ {0̄};
2  r ← inc(0̄);
3  while true do
4      if r ≠ ⊤ then
5          Let ℓ be maximal such that r = ⟨r⟩_ℓ;
6          S_r ← S_r ∪ ⋃_{1≤k≤(ℓ+1)/2} (CPre_O(S_{dec_{2k−1}(r)}) ∩ P_{2k−1});
7          repeat
8              S_r ← S_r ∪ (CPre_O(S_r) \ ⋃_{ℓ<k<c} P_k)
9          until a fixed-point for S_r is reached;
10     else if r = ⊤ then
11         S_⊤ ← S_⊤ ∪ ⋃_{1≤k≤⌊c/2⌋} (CPre_O(S_{dec_{2k−1}(⊤)}) ∩ P_{2k−1});
12         repeat
13             S_⊤ ← S_⊤ ∪ (CPre_O(S_⊤))
14         until a fixed-point for S_⊤ is reached;
15     r′ ← dec(r);
16     if S_{r′} ⊇ S_r and r < ⊤ then
17         r ← inc(r)
18     else if S_{r′} ⊇ S_r and r = ⊤ then
19         break
20     else
21         repeat
22             S_{r′} ← S_{r′} ∪ S_r;
23             r′ ← dec(r′);
24         until S_{r′} ⊇ S_r;
25         r ← inc(r′);
26 return V \ S_⊤
```

this information has to be propagated to its predecessors. This is achieved efficiently by maintaining *anti-monotonicity* among the sets, i.e., we have $S_{r'} \supseteq S_r$ for all r and all $r' < r$ before and after each iteration. Anti-monotonicity together with defining the sets $S_{r'}$ to contain vertices with rank *at least* r' instead of *exactly* r' enables us to decide whether the rank of a vertex v can be increased to r by only considering one set $S_{r'}$.

Structure of the algorithm. The set $S_{\bar{0}}$ is initialized with the set of all vertices V, while all other sets S_r for $r > \bar{0}$ are initially empty, i.e., the ranks of all vertices are initialized with the zero vector. The variable r is initially set to the second lowest rank $\mathrm{inc}(\bar{0})$ that is one at index 1 and zero otherwise. In the while-loop the set S_r is updated for the value of r at the beginning of the iteration (see below). After the update of S_r, it is checked whether the set corresponding to the next lowest rank already contains the vertices newly added to S_r, i.e., whether the anti-monotonicity is preserved. If the anti-monotonicity is preserved despite the update of S_r, then for $r < \top$ the value of r is increased to the next highest rank and for $r = \top$ the algorithm terminates. Otherwise the vertices newly added to S_r are also added to all sets with $r' < r$ that do not already contain them; the variable r is then updated to the lowest r' for which a new vertex is added to $S_{r'}$ in this iteration.

Update of set S_r. To reach a simultaneous fixed point of the lift-operators, the rank of a vertex v has to be increased to $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v)$ whenever the value of $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v)$ is strictly higher than $\rho_{\{S_r\}_r}(v)$ for the current ranking function $\rho_{\{S_r\}_r}$. Now consider a fixed

iteration of the while-loop and let r be as at the beginning of the while-loop. Let $\rho_{\{S_r\}_r}$ be denoted by ρ for short. In this update of the set S_r we want to add to S_r all vertices v with $\rho(v) < r$ and $\mathrm{Lift}(\rho, v)(v) \geq r$ under the condition that the priority of v allows v to be assigned the rank r, i.e., $r = \langle r \rangle_{\alpha(v)}$. Note that by the anti-monotonicity property the set S_r already contains all vertices with $\rho(v) \geq r$.

1. We first consider the case $r < \top$. Let ℓ be maximal such that $r = \langle r \rangle_\ell$, i.e., the first ℓ entries with indices 0 to $\ell - 1$ of r are 0 and the entry with index ℓ is larger than 0. Note that ℓ is odd. We have that only the $\mathrm{Lift}(., v)$-operators with $\alpha(v) \leq \ell$ can increase the rank of a vertex to r as all the others would set the element with index ℓ to 0.
   Recall that $\mathrm{Lift}(\rho, v)(v) = \mathrm{inc}_{\alpha(v)}(\mathrm{best}(\rho, v))$. The function best is implemented by the $\mathsf{CPre}_{\mathcal{O}}$ operator: For a player-$\mathcal{E}$ vertex the value of best increases only if the ranks of all successor have increased, for a player-$\mathcal{O}$ vertex it increases as soon as the maximum rank among the successor vertices has increased. The function $\mathrm{inc}_{\alpha(v)}(x)$ for $x < \top$ behaves differently for odd and even $\alpha(v)$ (see Section 2.3): If $\alpha(v)$ is odd, then $\mathrm{inc}_{\alpha(v)}(x)$ is the smallest rank y in M_h^∞ such that $y >_{\alpha(v)} x$, i.e., y is larger than x w.r.t. indices $\geq \alpha(v)$. If $\alpha(v)$ is even, then $\mathrm{inc}_{\alpha(v)}(x)$ is equal to x with the indices lower than $\alpha(v)$ set to 0.
   - **(i)** First, consider a $\mathrm{Lift}(\rho, v)$ operation with odd $\alpha(v) \leq \ell$, i.e., let $\alpha(v) = 2k - 1$ for some $1 \leq k \leq (\ell+1)/2$. Then $\mathrm{Lift}(\rho, v)(v) \geq r$ only if (a) $v \in V_{\mathcal{E}}$ and all successors w have $\rho(w) \geq \mathrm{dec}_{2k-1}(r)$, or (b) $v \in V_{\mathcal{O}}$ and one successor w has $\rho(w) \geq \mathrm{dec}_{2k-1}(r)$. That is, $\mathrm{Lift}(\rho, v)(v) \geq r$ only if $v \in \mathsf{CPre}_{\mathcal{O}}(S_{\mathrm{dec}_{2k-1}(r)})$. Vice versa, we have that if $v \in \mathsf{CPre}_{\mathcal{O}}(S_{\mathrm{dec}_{2k-1}(r)})$ then by $\rho = \rho_{\{S_r\}_r}$ also $\mathrm{Lift}(\rho, v)(v) \geq r$. This observation is implemented in SymbolicParityDominion in line 6, where such vertices v are added to S_r.
   - **(ii)** Now, consider a $\mathrm{Lift}(\rho, v)$ operation with even $\alpha(v) \leq \ell$, i.e., let $\alpha(v) = 2k$ for some $1 \leq k \leq \ell/2$. Then $\mathrm{Lift}(\rho, v)(v) \geq r$ only if
     - **(a)** $v \in V_{\mathcal{E}}$ and all successors w have $\rho(w) \geq r$, or
     - **(b)** $v \in V_{\mathcal{O}}$ and one successor w has $\rho(w) \geq r$.

     That is, $\mathrm{Lift}(\rho, v)(v) \geq r$ only if $v \in \mathsf{CPre}_{\mathcal{O}}(S_r)$. Vice versa, we have that if $v \in \mathsf{CPre}_{\mathcal{O}}(S_r)$ then $\mathrm{Lift}(\rho, v)(v) \geq r$. In SymbolicParityDominion these vertices are added iteratively in line 8 until a fixed point is reached. The algorithm also adds vertices v with odd priority to S_r, but due do the above argument we have $\mathrm{Lift}(\rho, v)(v) > r$ and thus they can be included in S_r.
2. The case $r = \top$ works similarly except that (a) every vertex is a possible candidate for being assigned the rank $\top$, independent of its priority (line 11), and (b) whenever x is equal to $\top$, $\mathrm{inc}_{\alpha(v)}(x)$ assigns the rank $\top$ independently of $\alpha(v)$ (line 13).

▶ **Example 2.** In this example we apply Algorithm SymbolicParityDominion to the parity game in Figure 1. We have $n_1 = 3$ and $n_3 = 1$ and thus we have to consider ranks in the co-domain $M_G^\infty = \{(0,0), (1,0), (2,0), (3,0), (0,1), (1,1), (2,1), (3,1), \top\}$ (we ignore entires of ranks that are always zero in this notation).

The algorithm initializes the set $S_{(0,0)}$ to $\{a, b, c, d, e, f, g, h\}$ and r to $(1, 0)$. All the other sets S_r are initialized as the empty set. It then proceeds as follows:

1. In the first iteration of the while-loop it processes $r = (1, 0)$. We have $\ell = 1$ and thus the only possible value of k in line 6 is $k = 1$. That is, line 6 adds the vertices in $\mathsf{CPre}_{\mathcal{O}}(S_{0,0}) \cap P_1 = \{a, c, h\}$ to $S_{(1,0)}$ and then in line 8 also b is added. We obtain $S_{(1,0)} = \{a, b, c, h\}$ and as $S_{(1,0)} \subseteq S_{(0,0)}$, the rank r is increased to $(2, 0)$.
2. In the second iteration it processes $r = (2, 0)$ and the vertex a is added to $S_{(2,0)}$ in line 6 and the vertex b is added to $S_{(2,0)}$ in line 8, i.e., $S_{(2,0)} = \{a, b\}$, and r is set to $(3, 0)$.

3. When processing $r = (3,0)$ the set $S_{(3,0)}$ is updated to $\{a, b\}$ and r is increased to $(0,1)$.
4. Now the algorithm processes the rank $(0,1)$ the first time. We have $\ell = 3$ and thus the possible values for k are 1 and 2. The vertex a is added to $S_{(0,1)}$ because it is contained in $\mathsf{CPre}_{\mathcal{O}}(S_{3,0}) \cap P_1$ and the vertex e is added because it is contained in $\mathsf{CPre}_{\mathcal{O}}(S_{0,0}) \cap P_3$ in line 6. Finally, also b and g are added in line 8. That is, we have $S_{(0,1)} = \{a, b, e, g\}$. Now as $S_{(3,0)} \not\subseteq S_{(0,1)}$, $S_{(2,0)} \not\subseteq S_{(0,1)}$ and $S_{(1,0)} \not\subseteq S_{(0,1)}$, we have to decrease r to $(1,0)$, and also to modify the other sets with smaller rank as follows: $S_{(1,0)} = \{a, b, c, e, g, h\}$; and $S_{(2,0)} = S_{(3,0)} = \{a, b, e, g\}$.
5. The algorithm considers $r = (1,0)$ again, makes no changes to $S_{(1,0)}$ and sets r to $(2,0)$.
6. Now considering $r = (2,0)$, the vertex h is added to the set $S_{(2,0)}$ in line 6, i.e., $S_{(2,0)} = \{a, b, e, g, h\}$, and, as h is already contained in $S_{(1,0)}$, r is increased to $(3,0)$.
7. The set $S_{(3,0)}$ is not changed and r is increased to $(0,1)$.
8. The set $S_{(0,1)}$ is not changed and r is increased to $(1,1)$.
9. The vertex a is added to $S_{(1,1)}$ in line 6 and the vertex b is added to $S_{(1,1)}$ in line 8, i.e., $S_{(1,1)} = \{a, b\}$, and r is increased to $(2,1)$.
10. The vertices a, b are added to $S_{(2,1)}$, i.e., $S_{(2,1)} = \{a, b\}$, and r is increased to $(3,1)$.
11. The vertices a, b are added to $S_{(3,1)}$, i.e., $S_{(3,1)} = \{a, b\}$, and r is increased to $\top$.
12. The vertex a is added to $S_\top$ in line 11 and b is added to $S_\top$ in line 13, i.e., $S_\top = \{a, b\}$. Now as $S_{(3,1)} \subseteq S_\top$, the algorithm terminates.

Finally we have that $S_{(0,0)} = \{a, b, c, d, e, f, g, h\}$, $S_{(1,0)} = \{a, b, c, e, g, h\}$, $S_{(2,0)} = \{a, b, e, g, h\}$, $S_{(3,0)} = S_{(0,1)} = \{a, b, e, g\}$, and $S_{(1,1)} = S_{(2,1)} = S_{(3,1)} = S_\top = \{a, b\}$, That is, the algorithm returns $\{c, d, e, f, g, h\}$ as the winning set of player $\mathcal{E}$. The final sets of the algorithm correspond to the progress measure ρ with $\rho(f) = \rho(d) = (0,0)$, $\rho(c) = (1,0)$, $\rho(h) = (2,0)$, $\rho(e) = \rho(g) = (0,1)$, and $\rho(a) = \rho(b) = \top$.

Sketch of bound on number of symbolic operations. Observe that each rank r is considered in at least one iteration of the while-loop but is only reconsidered in a later iteration if at least one vertex was added to the set S_r since the last time r was considered; in this case $O(c)$ one-step operations are performed. Thus the number of symbolic operations per set S_r is of the same order as the number of times a vertex is added to the set. Hence the algorithm can be implemented with $O(c \cdot n \cdot |M_h^\infty|)$ symbolic operations. For the co-domain $M_{\mathcal{G}}^\infty$ the bound $O(c \cdot n \cdot |M_{\mathcal{G}}^\infty|)$ is analogous.

Outline correctness proof. In the following proof we show that when Algorithm SymbolicParityDominion terminates, the ranking function $\rho_{\{S_r\}_r}$ is equal to the progress measure for the given parity game and the co-domain M_h^∞. The same proof applies to the co-domain $M_{\mathcal{G}}^\infty$. The algorithm returns the set of vertices that are assigned a rank $< \top$ when the algorithm terminates. By [39] this set is an $\mathcal{E}$-dominion that contains all $\mathcal{E}$-dominions of size at most $h + 1$ when the co-domain M_h^∞ is used, and by Lemma 1 this set is equal to the winning set of player $\mathcal{E}$ when the co-domain $M_{\mathcal{G}}^\infty$ is used. Thus it remains to show that $\rho_{\{S_r\}_r}$ equals the progress measure for the given co-domain when the algorithm terminates. We show that maintaining the following invariants over all iteration of the algorithm is sufficient for this and then prove that the invariants are maintained. All proofs are in Appendix A and are described for the co-domain M_h^∞.

▶ **Invariant 3** (Symbolic invariants). *In Algorithm SymbolicParityDominion the following three invariants hold. Every rank is from the co-domain M_h^∞ and the* $\mathrm{Lift}(., v)$*-operators are defined w.r.t. the co-domain. Let $\tilde{\rho}$ be the progress measure of the given parity game and let*

$\rho_{\{S_r\}_r}(v) = \max\{r \in M_h^\infty \mid v \in S_r\}$ *be the ranking function with respect to the sets* S_r *that are maintained by the algorithm.*

1. *Before and after each iteration of the while-loop we have that if a vertex* v *is in a set* S_{r_1} *then it is also in* S_{r_2} *for all* $r_2 < r_1$ *(anti-monotonicity).*
2. *Throughout Algorithm SymbolicParityDominion we have* $\tilde{\rho}(v) \geq \rho_{\{S_r\}_r}(v)$ *for all* $v \in V$.
3. *Before and after each iteration of the while-loop we have for the rank stored in* r *and all vertices* v *either* $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) \geq r$ *or* $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = \rho_{\{S_r\}_r}(v)$. *(b) After the update of* S_r *and before the update of* r *we additionally have* $v \in S_r$ *for all vertices* v *with* $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = r$ *(closure property).*

The intution behind the invariants is as follows. Invariant 3(1) ensures that the definition of the sets S_r and the ranking function $\rho_{\{S_r\}_r}$ is sound; Invariant 3(2) guarantees that $\rho_{\{S_r\}_r}$ is a lower bound on $\tilde{\rho}$ throughout the algorithm; and Invariant 3(3) shows that when the algorithm terminates, a fixed point of the ranking function $\rho_{\{S_r\}_r}$ with respect to the $\mathrm{Lift}(., v)$-operators is reached. Together these three properties guarantee that when the algorithm terminates the function $\rho_{\{S_r\}_r}$ corresponds to the progress measure, i.e., to the least simultaneous fixed point of the $\mathrm{Lift}(., v)$-operators. We prove the invariants by induction over the iterations of the while-loop. In particular, Invariant 3(1) is ensured by adding vertices newly added to a set S_r also to sets $S_{r'}$ with $r' < r$ that do not already contain them at the end of each iteration of the while-loop. For Invariant 3(2) we show that whenever $\rho_{\{S_r\}_r}(v)$ is increased, i.e., v is added to the set S_r, then no fixed point of the lift-operator for v was reached yet and thus also the progress measure for v has to be at least as high as the new value of $\rho_{\{S_r\}_r}(v)$. The intuition for the proof of Invariant 3(3) is as follows: We first show that $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = \rho_{\{S_r\}_r}(v)$ remains to hold for all vertices v for which the value of $\rho_{\{S_r\}_r}(v)$ is less than the smallest value r' for which $S_{r'}$ was updated in the considered iteration. In iterations in which the value of the variable r is not increased, this is already sufficient to show part (a) of the invariant. If r is increased, we additionally use part (b) to show part (a). For part (b) we prove by case analysis that, before the update of the variable r, a vertex with $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = r$ is included in S_r. The correctness of the algorithm then follows from the invariants as outlined above.

3.2 Reducing Space to Linear

Algorithm SymbolicParityDominion requires $|M_{\mathcal{G}}^\infty|$ many sets S_r, which is drastically beyond the space requirement of the progress measure algorithm for explicitly represented graphs. Thus we aim to reduce the space requirement to $O(n)$ many sets in a way that still allows to restore the sets S_r efficiently. For the sake of readability, we assume for this part that c is even. The main idea to reduce the space requirement is as follows.

1. Instead of storing sets S_r corresponding to a specific rank, we encode the value of each coordinate of the rank r separately. That is, we define the sets $C_0^i, C_1^i \ldots, C_{n_i}^i$ for each odd priority i. Intuitively, a vertex is in the set C_x^i iff the i-th coordinate of the rank of v is x. Given these $O(c + n) \in O(n)$ sets, we have encoded the exact rank vector r of each vertex with $r < \top$. To also cover vertices with rank $\top$, we additionally store the set $S_\top$.
2. Whenever the algorithm needs to process a set S_r, we reconstruct it from the stored sets, using a linear number of set operations. Algorithm SymbolicParityDominion has to be adapted as follows. First, at the beginning of each iteration we have to compute the set S_r and up to $c/2$ sets $S_{r'}$ that correspond to some predecessor r' of r. Second, at the end of each iteration we have to update the sets C_x^i to incorporate the updated set S_r.

Computing a set S_r from the sets C_x^i. Let r_i denote the i-th entry of r. To obtain the set $S_r^=$ of vertices with rank *exactly* r (for $r < \top$), one can simply compute the intersection $\bigcap_{1 \le k \le c/2} C^{2k-1}_{r_{2k-1}}$ of the corresponding sets C_x^i. However, in the algorithm we need the sets S_r containing all vertices v with a rank at least r and computing all sets $S_{r'}^=$ with $r' \ge r$ is not efficient. Towards a more efficient method to compute S_r, recall that a rank $r' < \top$ is higher than r if either (a) the right most odd element of r' is larger than the corresponding element in in r, i.e., $r'_{c-1} > r_{c-1}$, or (b) if r and r' coincide on the i right most odd elements, i.e., $r'_{c-2k+1} = r_{c-2k+1}$ for $1 \le k \le i$, and $r'_{c-2i-1} > r_{c-2i-1}$. In case (a) we can compute the corresponding vertices by $S_r^0 = \bigcup_{r_{c-1} < x \le n_{c-1}} C_x^{c-1}$ while in case (b) we can compute the corresponding vertices by $S_r^i = \bigcap_{1 \le k \le i} C^{c-2k+1}_{r_{c-2k+1}} \cap \bigcup_{r_{c-2i-1} < x \le n_{c-2i-1}} C_x^{c-2i-1}$ for $1 \le i \le c/2 - 1$. That is, we can reconstruct the set S_r by the following union of the above sets S_r^i, the set $S_r^=$ of vertices with rank r, and the set $S_\top$ of vertices with rank $\top$:

$$S_r = S_\top \cup S_r^= \cup \bigcup_{i=0}^{c/2-1} S_r^i$$

Hence, a set S_r can be computed with $O(c+n) \in O(n)$ many $\cup$ and $O(c)$ many $\cap$ operations; for the latter bound we use an additional set to store the set $\bigcap_{1 \le k \le i} C^{c-2k+1}_{r_{c-2k+1}}$ for the current value of i, such that for each set S_r^i we just need two $\cap$ operations. This implies the following lemma.

▶ **Lemma 4.** *Given the sets C_x^i as defined above, we can compute the set S_r that contains all vertices with rank at least r with $O(n)$ many symbolic set operations. No symbolic one-step operation is needed.*

Updating a set S_r. Now consider we have updated a set S_r during the iteration of the while-loop, and now we want to store the updated set S_r within the sets C_x^i. That is, we have already computed the fixed-point for S_r and are now in line 15 of the Algorithm. To this end, let S_r^{old} be the set as stored in C_x^i and S_r^{new} the updated set, which is a superset of the old one. First, one computes the difference S_r^{diff} between the two sets $S_r^{\text{diff}} = S_r^{\text{new}} \setminus S_r^{\text{old}}$; intuitively, the set S_r^{diff} contains the vertices for which the algorithm has increased the rank. Now for the vertices of S_r^{diff} we have to

(i) delete their old values by updating C_x^i to $C_x^i \setminus S_r^{\text{diff}}$ for each $i \in \{1, 3, \dots, c-1\}$ and each $x \in \{0, \dots, n_i\}$ and

(ii) store the new values by updating C_x^i to $C_x^i \cup S_r^{\text{diff}}$ for $i \in \{1, 3, \dots, c-1\}$ and $x = r_i$.

In total we have $O(c)$ many $\cup$ and $O(n)$ many $\setminus$ operations.

Notice that the update operation for a set S_r, as described above, also updates all sets $S_{r'}$ for $r' < r$. Thus, when using the more succinct representation via the sets C_x^i and executing SymbolicParityDominion literally, the computation of the maximal rank r' s.t. $S_{r'} \supseteq S_r$ would fail because of the earlier update of S_r. Hence, we have to postpone the update of S_r till the end of the iteration and adjust the computation of r' as follows. We do not update the set S'_r, and first compute the final value for r' by decrementing r' until $S_{r'} \supseteq S_r$ and then update S_r to S_r^{new} and thus implicitly also update the sets $S_{\tilde{r}}$ to $S_{\tilde{r}} \cup S_r$ for $r' < \tilde{r} < r$. This gives the following lemma.

▶ **Lemma 5.** *In each iteration of SymbolicParityDominion only $O(n)$ symbolic set operations are needed to update the sets C_x^i, and no symbolic one-step operation is needed.*

Number of Set Operations. To sum up, when introducing the succinct representation of the sets S_r, we only need additional $\cup$, $\cap$, and $\setminus$ operations, while the number of CPre_z

operations is unchanged. We show in Appendix A that whenever the algorithm computes or updates a set S_r, then we can charge a CPre_z operation for it, and each CPre_z operation is only charged for a constant number of set computations and updates. Hence, as both computing a set S_r and updating the sets C_x^i can be done with $O(n)$ set operations, the number of the additional set operations in SymbolicParityDominion is in $O(n \cdot \#\mathsf{CPre})$, for $\#\mathsf{CPre}$ being the number of CPre_z operations in the algorithm.

Putting Things Together. We presented a set-based symbolic implementation of the progress measure that uses $O(n)$ sets, $O(c \cdot n \cdot |M_h^\infty|)$ symbolic one-step operations and at most a factor of n more symbolic set operations. Using that $|M_h^\infty| \leq \binom{h+\lfloor c/2 \rfloor}{h} + 1$ we obtain the following key lemma that summarizes the result for computing dominions.

▶ **Key Lemma 6.** *For a parity game with n vertices and c priorities, and $h \in [1, n-1]$, SymbolicParityDominion computes a player-$\mathcal{E}$ dominion that contains all $\mathcal{E}$-dominions with at most $h+1$ vertices and can be implemented with $O\left(c \cdot n \cdot \binom{h+\lfloor c/2 \rfloor}{h}\right)$ symbolic one-step operations, $O\left(c \cdot n^2 \cdot \binom{h+\lfloor c/2 \rfloor}{h}\right)$ symbolic set operations, and $O(n)$ many sets.*

To solve parity games directly with Algorithm SymbolicParityDominion, we use the co-domain $M_{\mathcal{G}}^\infty$ instead of M_h^∞. Recall that we have $|M_{\mathcal{G}}^\infty| \in O\big(\big(\frac{n}{\lfloor c/2 \rfloor}\big)^{\lfloor c/2 \rfloor}\big)$ [29].

▶ **Theorem 7.** *Let $\xi(n,c) = \big(\frac{n}{\lfloor c/2 \rfloor}\big)^{\lfloor c/2 \rfloor}$. Algorithm SymbolicParityDominion computes the winning sets of parity games and can be implemented with $O\big(c \cdot n \cdot \xi(n,c)\big)$ symbolic one-step operations, $O\big(c \cdot n^2 \cdot \xi(n,c)\big)$ symbolic set operations, and $O(n)$ many sets.*

See the full version [12] for how to construct winning strategies within the same bounds.

4 Extensions and Conclusion

Big-Step Algorithm. We presented a set-based symbolic algorithm for computing a progress measure that solves parity games. Since the progress measure algorithm can also compute dominions of bounded size, it can be combined with the big step approach of [39] to improve the number of symbolic steps as stated in the following theorem.

▶ **Theorem 8.** *Let $\gamma(c) = c/3 + 1/2 - 4/(c^2-1)$ for odd c and $\gamma(c) = c/3 + 1/2 - 1/(3c) - 4/c^2$ for even c. There is a symbolic Big Step Algorithm that computes the winning sets for parity games and with the minimum of $O(n \cdot (\kappa \cdot n/c)^{\gamma(c)})$, for some constant κ, and $n^{O(\sqrt{n})}$ symbolic one-step operations and stores only $O(n)$ many sets.*

Concluding Remarks. In this work we presented improved set-based symbolic algorithms for parity games, and equivalently modal μ-calculus model checking. Our main contribution improves the symbolic algorithmic complexity of one of the most fundamental problems in the analysis of program logics, with numerous applications in program analysis and reactive synthesis. There are several practical approaches to solve parity games, such as, [15, 23, 27, 26, 3] and [42]. A practical direction of future work would be to explore whether our algorithmic ideas can be complemented with engineering efforts to obtain scalable symbolic algorithms for reactive synthesis of systems. An interesting theoretical direction of future work is to obtain set-based symbolic algorithms for parity games with quasi-polynomial complexity. The breakthrough result of [10] (see also [24]) relies on alternating poly-logarithmic space Turing machines. The follow-up papers of [30] and [21] that slightly improve the running

time and reduce the space complexity from quasi-polynomial to quasi-linear rely on succinct notions of progress measures. All these algorithms are non-symbolic, and symbolic versions of these algorithms are an open question, in particular encoding the novel succinct progress measures in the symbolic setting when storing at most a linear number of sets.

References

1 S. B. Akers. Binary decision diagrams. *IEEE Trans. Comput.*, C-27(6):509–516, 1978.

2 R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *JACM*, 49:672–713, 2002. doi:10.1145/585265.585270.

3 M. Benerecetti, D. Dell'Erba, and F. Mogavero. Solving parity games via priority promotion. In *CAV*, pages 270–290, 2016. doi:10.1007/978-3-319-41540-6_15.

4 T. A. Beyene, S. Chaudhuri, C. Popeea, and A. Rybalchenko. A constraint-based approach to solving games on infinite graphs. In *POPL*, pages 221–234, 2014.

5 H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithms for parity games. In *STACS*, pages 663–674, 2003. doi:10.1007/3-540-36494-3_58.

6 A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. R. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *TCS*, 178(1-2):237–255, 1997.

7 R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, C-35(8):677–691, 1986. doi:10.1109/TC.1986.1676819.

8 J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. AMS*, 138:295–311, 1969.

9 D. Bustan, O. Kupferman, and M. Y. Vardi. A measured collapse of the modal μ-calculus alternation hierarchy. In *STACS*, pages 522–533, 2004.

10 C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. In *STOC*, 2017. To appear.

11 P. Cerný, K. Chatterjee, T. A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for concurrent programs. In *CAV*, pages 243–259, 2011.

12 K. Chatterjee, W. Dvořák, M. Henzinger, and V. Loitzenbauer. Improved set-based symbolic algorithms for parity games. *CoRR*, abs/1706.04889, 2017. URL: https://arxiv.org/abs/1706.04889.

13 K. Chatterjee, M. Henzinger, and V. Loitzenbauer. Improved Algorithms for One-Pair and k-Pair Streett Objectives. In *LICS*, pages 269–280, 2015. doi:10.1109/LICS.2015.34.

14 A. Church. Logic, arithmetic, and automata. In *ICM*, pages 23–35, 1962.

15 L. de Alfaro and M. Faella. An accelerated algorithm for 3-color parity games with an application to timed games. In *CAV*, pages 108–120, 2007.

16 L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR*, pages 142–156, 2003.

17 L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *EMSOFT*, pages 148–165. 2001. doi:10.1007/3-540-45449-7_11.

18 L. de Alfaro, T. A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR*, pages 536–550, 2001. doi:10.1007/3-540-44685-0_36.

19 E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS*, pages 368–377, 1991. doi:10.1109/SFCS.1991.185392.

20 E. A. Emerson and Ch.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *LICS*, pages 267–278, 1986.

21 J. Fearnley, S. Jain, S. Schewe, F. Stephan, and D. Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. To be announced at SPIN, 2017. URL: http://arxiv.org/abs/1703.01296.

22 O. Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *LICS*, pages 145–156, 2009. doi:10.1109/LICS.2009.27.

23 O. Friedmann and M. Lange. Solving parity games in practice. In *ATVA*, pages 182–196, 2009.

24 H. Gimbert and R. Ibsen-Jensen. A short proof of correctness of the quasi-polynomial time algorithm for parity games, 2017. URL: https://arxiv.org/abs/1702.01953.

25 T. A. Henzinger, O. Kupferman, and S. K. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002. doi:10.1006/inco.2001.3085.

26 P. Hoffmann and M. Luttenberger. Solving parity games on the GPU. In *ATVA*, pages 455–459, 2013.

27 M. Huth, J. Huan-Pu Kuo, and N. Piterman. Concurrent small progress measures. In *HVC*, pages 130–144, 2011.

28 B. Jobstmann, A. Griesmayer, and R. Bloem. Program repair as a game. In *CAV*, pages 226–238, 2005. doi:10.1007/11513988_23.

29 M. Jurdziński. Small Progress Measures for Solving Parity Games. In *STACS*, pages 290–301, 2000. doi:10.1007/3-540-46541-3_24.

30 M. Jurdziński and R. Lazić. Succinct progress measures for solving parity games. To be announced at LICS, 2017. URL: https://arxiv.org/abs/1702.05051.

31 M. Jurdziński, M. Paterson, and U. Zwick. A Deterministic Subexponential Algorithm for Solving Parity Games. *SIAM J. Comput.*, 38(4):1519–1532, 2008.

32 C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell System Techn. J.*, 38(4):985–999, 1959. doi:10.1002/j.1538-7305.1959.tb01585.x.

33 P. Madhusudan and P. S. Thiagarajan. Distributed controller synthesis for local specifications. In *ICALP*, pages 396–407, 2001. doi:10.1007/3-540-48224-5_33.

34 R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993. doi:10.1016/0168-0072(93)90036-D.

35 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989. doi:10.1145/75277.75293.

36 P. J. Ramadge and W. Murray Wonham. Supervisory control of a class of discrete-event processes. *SIAM J. Control Optim.*, 25(1):206–230, 1987. doi:10.1137/0325013.

37 S. Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327, 1988.

38 S. Safra. *Complexity of automata on infinite objects*. PhD thesis, Weizmann Inst., 1989.

39 S. Schewe. Solving Parity Games in Big Steps. *JCSS*, 84:243–262, 2017.

40 H. Seidl. Fast and simple nested fixpoints. *IPL*, 59(6):303–308, 1996.

41 F. Somenzi. CUDD: CU decision diagram package release 3.0.0, 2015. URL: http://vlsi.colorado.edu/~fabio/CUDD/.

42 S. Vester. Winning cores in parity games. In *LICS*, pages 662–671, 2016.

43 J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV*, pages 202–215, 2000. doi:10.1007/10722167_18.

44 W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.

A Technical Appendix: Details for Section 3

Correctness. The correctness of Algorithm SymbolicParityDominion, stated in the following lemma, follows from combining Lemma 10 with Lemmata 11–13, which we prove below.

▶ **Lemma 9** (Correctness). *Algorithm SymbolicParityDominion computes the progress measure for a given parity game (with n vertices) and a given set of possible ranks M_h^∞ (for some integer $h \in [1, n-1]$).*

▶ **Lemma 10.** *Assuming that Invariant 3 holds, the ranking function $\rho_{\{S_r\}_r}$ induced by the family of sets $\{S_r\}_r$ at termination of Algorithm SymbolicParityDominion is equal to the progress measure for the given parity game and the co-domain M_h^∞.*

Proof. Recall that the progress measure is the least simultaneous fixed point of all $\mathrm{Lift}(., v)$-operators for the given parity game (where inc, dec, and the ordering of ranks are w.r.t. the given co-domain) and let the progress measure be denoted by $\tilde{\rho}$. Let $\{S_r\}_r$ be the sets in the algorithm at termination. For all $v \in V$ the ranking function $\rho_{\{S_r\}_r}(v)$ is defined as $\max\{r \in M_h^\infty \mid v \in S_r\}$. By Invariant 3(2) we have $\rho_{\{S_r\}_r}(v) \leq \tilde{\rho}(v)$ for all $v \in V$.

When the algorithm terminates, with $r = \top$, we have by Invariant 3(3) $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = \rho_{\{S_r\}_r}(v)$ for each vertex v and thus $\rho_{\{S_r\}_r}$ is a simultaneous fixed point of the $\mathrm{Lift}(., v)$-operators. Now, as $\tilde{\rho}$ is the least simultaneous fixed point of all $\mathrm{Lift}(., v)$-operators, we obtain $\rho_{\{S_r\}_r}(v) \geq \tilde{\rho}(v)$ for all $v \in V$. Hence we have $\rho_{\{S_r\}_r}(v) = \tilde{\rho}(v)$ for all $v \in V$. ◀

▶ **Lemma 11.** *Before and after each iteration of the while-loop in Alg. SymbolicParityDominion we have $S_{r_1} \supseteq S_{r_2}$ for all $r_1 \leq r_2$ with $r_1, r_2 \in M_h^\infty$, i.e., Invariant 3(1) holds.*

Proof. The proof is by induction over the iterations of the while-loop. The claim is satisfied when we first enter the while-loop and only $S_{\bar{0}}$ is non-empty. It remains to show that when the claim is valid at the beginning of a iteration then the claim also hold afterwards. By the induction hypothesis, the sets $S_{r'}$ for $r' < r$ are monotonically decreasing. Thus it is sufficient to find the lowest rank r^* such that for all $r^* \leq r' < r$ we have $S_r \not\subseteq S_{r'}$ and add the vertices newly added to S_r to the sets $S_{r'}$ with $r^* \leq r' < r$, which is done in lines 16–25 of the while-loop. ◀

▶ **Lemma 12.** *Let $\tilde{\rho}$ be the progress measure of the given parity game and let $\rho_{\{S_r\}_r}(v) = \max\{r \in M_h^\infty \mid v \in S_r\}$ be the ranking function with respect to the family of sets $\{S_r\}_r$ that is maintained by the algorithm. Throughout Algorithm SymbolicParityDominion we have $\tilde{\rho}(v) \geq \rho_{\{S_r\}_r}(v)$ for all $v \in V$, i.e., Invariant 3(2) holds.*

Proof. We show the lemma by induction over the iterations of the while-loop. Before the first iteration of the while-loop only $S_{\bar{0}}$ is non-empty, thus the claim holds by $\tilde{\rho} \geq \bar{0}$.

Assume we have $\rho_{\{S_r\}_r}(v) \leq \tilde{\rho}(v)$ for all $v \in V$ before an iteration of the while-loop. We show that $\rho_{\{S_r\}_r}(v) \leq \tilde{\rho}(v)$ also holds during and after the iteration of the while-loop. As the update of $S_{r'}$ in line 22 does not change $\rho_{\{S_r\}_r}$, we only have to show that the invariant is maintained by the update of S_r in lines 4–14. Further $\rho_{\{S_r\}_r}(v)$ only changes for vertices newly added to S_r, thus we only have to take these vertices into account.

Let ℓ be the maximal index such that $r = \langle r \rangle_\ell$ or the highest odd priority if $r = \top$. Assume $r < \top$, the argument for $r = \top$ is analogous. The algorithm adds vertices to S_r in (1) line 6 and (2) line 8. In case (1) we add the vertices $\bigcup_{1 \leq k \leq (\ell+1)/2} (\mathsf{CPre}_{\mathcal{O}}(S_{\mathrm{dec}_{2k-1}(r)}) \cap P_{2k-1})$ to S_r. Let $v \in \mathsf{CPre}_{\mathcal{O}}\ (S_{\mathrm{dec}_{2k-1}(r)}) \cap P_{2k-1}$ for some $1 \leq k \leq (\ell+1)/2$.

- If $v \in V_{\mathcal{E}} \cap P_{2k-1}$, then all successors w of v are in $S_{\mathrm{dec}_{2k-1}(r)}$ and thus, by the induction hypothesis, have $\tilde{\rho}(w) \geq \mathrm{dec}_{2k-1}(r)$. Now as $v \in P_{2k-1}$, it has rank $\tilde{\rho}(v)$ at least $\mathrm{inc}_{2k-1}(\mathrm{dec}_{2k-1}(r)) = r$.
- If $v \in V_{\mathcal{O}} \cap P_{2k-1}$, at least one successors w of v is in $S_{\mathrm{dec}_{2k-1}(r)}$ and thus, by the induction hypothesis, has $\tilde{\rho}(w) \geq \mathrm{dec}_{2k-1}(r)$. Now as $v \in P_{2k-1}$, it has rank $\tilde{\rho}(v)$ at least $\mathrm{inc}_{2k-1}(\mathrm{dec}_{2k-1}(r)) = r$.

For case (2) consider a vertex $v \in \mathsf{CPre}_{\mathcal{O}}(S_r) \setminus \bigcup_{\ell < k \leq d} P_k$ added in line 8.

- If $v \in V_{\mathcal{E}}$, all successors w of v are in S_r and thus, by the induction hypothesis, have $\tilde{\rho}(w) \geq r$. Since the priority of v is $\leq \ell$, we have $\tilde{\rho}(v) \geq \langle r \rangle_\ell = r$.

- If $v \in V_{\mathcal{O}}$, at least one successors w of v is in S_r and thus, by the induction hypothesis, has $\tilde{\rho}(w) \geq r$. Since the priority of v is $\leq \ell$, we have $\tilde{\rho}(v) \geq \langle r \rangle_\ell = r$. ◀

▶ **Lemma 13.** *Before and after each iteration of the while loop we have for the rank stored in r and all vertices v either* $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) \geq r$ *or* $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = \rho_{\{S_r\}_r}(v)$. *At line 15 of the algorithm we additionally have $v \in S_r$ for all vertices v for which the value of* $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v)$ *is equal to r. Thus Invariant 3(3) holds.*

Proof. We show the claim by induction over the iterations of the while-loop. Before we first enter the loop, we have $r = \mathrm{inc}(\bar{0})$ and $S_{\bar{0}} = V$ and thus the claim is satisfied. For the inductive step, let r^{old} be the value of r and ρ^{old} the ranking function $\rho_{\{S_r\}_r}$ before a fixed iteration of the while-loop and assume we have for all $v \in V$ either $\mathrm{Lift}(\rho^{\mathrm{old}}, v)(v) \geq r^{\mathrm{old}}$ or $\mathrm{Lift}(\rho^{\mathrm{old}}, v)(v) = \rho^{\mathrm{old}}(v)$ before the iteration of the while-loop. Let r^{new} be the value of r and ρ^{new} the ranking function $\rho_{\{S_r\}_r}$ after the iteration. We have three cases for the value of r^{new}:

1. $r^{\mathrm{new}} = \mathrm{inc}(r^{\mathrm{old}})$ (line 16),
2. $r^{\mathrm{new}} = r^{\mathrm{old}} = \top$ (line 18), or
3. $r^{\mathrm{new}} < r^{\mathrm{old}}$, i.e., the rank is decreased in lines 21–25 to maintain anti-monotonicity.

We show in Claim 14 that, in all three cases, if a set $S_{r'}$, for some $r' < r^{\mathrm{old}}$, is not changed in the considered iteration of the while-loop then for all $v \in V$ with $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) \leq r'$ we have that $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) = \rho^{\mathrm{new}}(v)$.

Given Claim 14, we prove the first part of the invariant as follows. In the case (1) the lowest (and only) rank for which the set is updated is r^{old}, thus it remains to show $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) = \rho^{\mathrm{new}}(v)$ for vertices with $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) = r^{\mathrm{old}}$, which is done by showing the second part of the invariant, namely that $v \in S_{r^{\mathrm{old}}}$ for all vertices v with $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = r^{\mathrm{old}}$ after the update of the set $S_{r^{\mathrm{old}}}$ in lines 4–14; for case (1) we have $\rho^{\mathrm{new}} = \rho_{\{S_r\}_r}$ at this point.

In the cases (2) and (3) we have that the lowest rank for which the set is updated in the iteration is equal to r^{new}, thus Claim 14 implies that the invariant $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) \geq r^{\mathrm{new}}$ or $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) = \rho^{\mathrm{new}}(v)$ holds for all $v \in V$ after the while-loop.

▶ **Claim 14.** *Let $r^* \leq r^{old}$ be a rank with the guarantee that no set corresponding to a lower rank than r^* is changed in this iteration of the while-loop. Then we have for all $v \in V$ either* $\mathrm{Lift}(\rho^{new}, v)(v) \geq r^*$ *or* $\mathrm{Lift}(\rho^{new}, v)(v) = \rho^{new}(v)$ *after the iteration of the while-loop.*

To prove the claim, note that since each set S_r is monotonically non-decreasing over the algorithm, we have $\rho^{\mathrm{new}}(v) \geq \rho^{\mathrm{old}}(v)$ and $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) \geq \rho^{\mathrm{new}}(v)$. Assume by contradiction that there is a vertex v with $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) > \rho^{\mathrm{new}}(v)$ and $\mathrm{Lift}(\rho^{\mathrm{new}}, v)(v) < r^*$. The latter implies $\mathrm{best}(\rho^{\mathrm{new}}, v) < r^*$. By the induction hypothesis for $r^* \leq r^{\mathrm{old}}$ we have $\mathrm{Lift}(\rho^{\mathrm{old}}, v)(v) = \rho^{\mathrm{old}}(v)$. By $\rho^{\mathrm{new}}(v) \geq \rho^{\mathrm{old}}(v)$ and the definition of the lift-operator this implies $\mathrm{best}(\rho^{\mathrm{new}}, v) > \mathrm{best}(\rho^{\mathrm{old}}, v)$, i.e., the rank assigned to at least one vertex w with $(v, w) \in E$ is increased. By $\mathrm{best}(\rho^{\mathrm{new}}, v) < r^*$ this implies that a set $S_{r'}$ with $r' < r^*$ is changed in this iteration, a contradiction to the definition of r^*. Hence, the claim holds.

It remains to show that $v \in S_{r^{\mathrm{old}}}$ for all vertices v with $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = r^{\mathrm{old}}$ after the update of the set $S_{r^{\mathrm{old}}}$ in lines 4–14. Towards a contradiction assume that there is a $v \notin S_{r^{\mathrm{old}}}$ such that $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = r^{\mathrm{old}}$. Assume $v \in V_{\mathcal{E}}$, the argument for $v \in V_{\mathcal{O}}$ is analogous. Let ℓ be maximal such that $r^{\mathrm{old}} = \langle r^{\mathrm{old}} \rangle_\ell$ for $r^{\mathrm{old}} < \top$ and let ℓ be the highest odd priority in the parity game for $r^{\mathrm{old}} = \top$. Notice that $\alpha(v)$ can be at most ℓ for $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = r^{\mathrm{old}}$ to hold. We now distinguish two cases depending on whether $\alpha(v)$ is odd or even.

- If $\alpha(v)$ is odd, i.e., $\alpha(v) = 2k-1$ for some $k \leq (\ell+1)/2$, then we have that all successors w of v have $\rho_{\{S_r\}_r}(w) \geq \mathrm{dec}_{2k-1}(r^{\mathrm{old}})$ and thus, by Lemma 11, $w \in S_{\mathrm{dec}_{2k-1}(r^{\mathrm{old}})}$. But then v would have being included in $S_{r^{\mathrm{old}}}$ in line 6, a contradiction.
- If $\alpha(v)$ is even, i.e., $\alpha(v) = 2k$ for some $k \leq \ell/2$, then we have that all successors w of v have $\rho_{\{S_r\}_r}(w) \geq r^{\mathrm{old}}$. Then by the definition of $\rho_{\{S_r\}_r}$ it must be that $w \in S_{r'}$ for some $r' \geq r^{\mathrm{old}}$ and by Lemma 11 it must be that $w \in S_{r^{\mathrm{old}}}$. But then v would have being included in $S_{r^{\mathrm{old}}}$ in line 8, a contradiction.

Thus, after the update of the set $S_{r^{\mathrm{old}}}$ we have that $v \in S_{r^{\mathrm{old}}}$ for all vertices v with $\mathrm{Lift}(\rho_{\{S_r\}_r}, v)(v) = r^{\mathrm{old}}$. Together we the above observations this proves the lemma. ◀

Number of symbolic operations. We address next the number of symbolic operations of Algorithm SymbolicParityDominion when using the sets S_r directly. We analyze the number of symbolic operations when using a linear number of sets below. The main idea is that a set S_r is only reconsidered if at least one new vertex was added to S_r.

▶ **Lemma 15.** *For parity games with n vertices and c priorities Algorithm SymbolicParityDominion takes $O(c \cdot n \cdot |M_h^\infty|)$ many symbolic operations and uses $O(|M_h^\infty|)$ many sets, where h is some integer in $[1, n-1]$.*

Proof. In the algorithm we use one set S_r for each $r \in M_h^\infty$ and thus $|M_h^\infty|$ many sets. We first consider the number of symbolic operations needed to compute the sets S_r in lines 4–14, and then the number of symbolic operations to compute the new value of r in lines 15–25.

1. Whenever we consider a set S_r, we first initialize the set with $O(c)$ many symbolic operations (lines 6 & 11). After that we do a fixed-point computation that needs symbolic operations proportional to the number of added vertices. Now fix a set S_r and consider all the fixed-point computations for S_r over the whole algorithm. As only $O(n)$ many vertices can be added to S_r, all these fixed-points can be computed in $O(n+\#r)$ symbolic operations, where $\#r$ is the number of times the set S_r is considered (the algorithm needs a constant number of symbolic operations to realize that a fixed-point was already reached). Each set S_r is considered at least once and only reconsidered when some new vertices are added to the set, i.e., it is considered at most n times. Thus for each set S_r we have $O(c \cdot n)$ many operations, which gives a total number of operations of $O(c \cdot n \cdot |M_h^\infty|)$.
2. Now consider the computation of the new value of r in lines 15–25. Lines 15–19 take a constant number of operations. It remains to count the iterations of the repeat-until loop in lines 20–25, which we bound by the number of iterations of the while-loop as follows. Whenever a set $S_{r'}$ is considered as the left side argument in line 24, then the new value for r is less or equal to $\mathrm{inc}(r')$ and thus there will be another iteration of the while-loop considering $\mathrm{inc}(r')$. As there are only $O(n \cdot |M_h^\infty|)$ many iterations of the while-loop over the whole algorithm, there are only $O(n \cdot |M_h^\infty|)$ many iterations of the repeat-until loop in total. In each iteration a constant number of operations is performed.

By 1. and 2. we have that Algorithm SymbolicParityDominion takes $O(c \cdot n \cdot |M_h^\infty|)$ many symbolic operations. ◀

Number of set operations in linear space algorithm. For the proof of Lemma 6 and thus of Theorem 7 it remains to show that whenever the algorithm computes or updates a set S_r using the succinct representation with the sets C_x^i introduced in Section 3.2, then we can charge a CPre_z operation for it, and each CPre_z operation is only charged for a constant number of set computations and updates. The argument is as follows.

(a) Whenever the algorithm computes a set S_r in line 6 or 11, at least one CPre_z computation with this set is done.

(b) Now consider the computation of the new value of r. The subset tests in lines 16 and 18 are between a set that was already computed in line 6 or 11 and the set computed in line 8 or 13 and thus, if we store these sets, we do not require additional operations. Whenever a set $S_{r'}$ is considered as the left side argument in line 24, then the new value of r is less or equal to $\mathrm{inc}(r')$ and thus there will be another iteration of the while-loop considering $\mathrm{inc}(r')$. Hence, we can charge the additional operations needed for the comparison to the CPre_z operations of the next iteration that processes the rank $\mathrm{inc}(r')$.

(c) Finally, we only need to update the sets C^i_x once per iteration and in each iteration we perform at least one CPre_z computation that we can charge for the update.

Slicewise Definability in First-Order Logic with Bounded Quantifier Rank*

Yijia Chen[1], Jörg Flum[2], and Xuangui Huang[3]

1 **School of Computer Science, Fudan University, Shanghai, China**
yijiachen@fudan.edu.cn
2 **Mathematisches Institut, Universität Freiburg, Germany**
joerg.flum@math.uni-freiburg.de
3 **Department of Computer Science, Shanghai Jiao Tong University, China**
stslxg@gmail.com

Abstract

For every $q \in \mathbb{N}$ let FO_q denote the class of sentences of first-order logic FO of quantifier rank at most q. If a graph property can be defined in FO_q, then it can be decided in time $O(n^q)$. Thus, minimizing q has favorable algorithmic consequences. Many graph properties amount to the existence of a certain set of vertices of size k. Usually this can only be expressed by a sentence of quantifier rank at least k. We use the color coding method to demonstrate that some (hyper)graph problems can be defined in FO_q where q is independent of k. This property of a graph problem is equivalent to the question of whether the corresponding parameterized problem is in the class para-AC^0.

It is crucial for our results that the FO-sentences have access to built-in addition and multiplication (and constants for an initial segment of natural numbers whose length depends only on k). It is known that then FO corresponds to the circuit complexity class uniform AC^0. We explore the connection between the quantifier rank of FO-sentences and the depth of AC^0-circuits, and prove that $\mathrm{FO}_q \subsetneq \mathrm{FO}_{q+1}$ for structures with built-in addition and multiplication.

1998 ACM Subject Classification F.1.1 [Models of Computation] Unbounded-Action Devices), F.1.3 [Complexity Measures and Classes] Complexity Hierarchies, F.4.1 [Mathematical Logic] Computational logic

Keywords and phrases first-order logic, quantifier rank, parameterized AC^0, circuit depth

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.19

1 Introduction

Many graph problems ask, given a graph $\mathcal{G}$ and a natural number k, for a set C of vertices of $\mathcal{G}$ of size k with a certain property. A well-known example is the *vertex cover problem* where the set C is required to have the property that every edge in $\mathcal{G}$ has at least one end in C. The set C is then called a *vertex cover* of $\mathcal{G}$ of size k. It is routine to show that the vertex cover problem is not in the complexity class AC^0. However, its parameterized version p-Vertex-Cover, that is,

p-Vertex-Cover	
Input:	A graph $\mathcal{G}$.
Parameter:	k.
Question:	Does $\mathcal{G}$ have a vertex cover of size k?

* This research is partially supported by the Sino-German Center for Research Promotion (CDZ 996) and National Nature Science Foundation of China (Project 61373029).

is in the class para-AC^0 [7], the parameterized version of AC^0.

If in p-Vertex-Cover we fix the parameter k, we get the *kth slice* of the problem. Clearly, the existence of a vertex cover of size k can be expressed by the following sentence of first-order logic FO

$$\psi_k := \exists x_1 \cdots \exists x_k \Big(\bigwedge_{1 \le i < j \le k} x_i \neq x_j \wedge \forall u \forall v \big(Euv \to \bigvee_{i=1}^{k} (u = x_i \vee v = x_i) \big) \Big). \tag{1}$$

In other words, a graph $\mathcal{G}$ is in the kth slice of p-Vertex-Cover if and only if $\mathcal{G}$ satisfies ψ_k. Observe that the quantifier rank $\mathrm{qr}(\psi_k)$ of ψ_k, the maximum nested depth of quantifiers in ψ_k, is $k+2$. Hence the naive algorithm derived from ψ_k has running time $O(|\mathcal{G}|^{k+2})$. Clearly it is far worse than the existing linear time algorithms for deciding whether a graph contains a vertex cover of size k. An immediate question is whether the kth slice can be defined by a sentence of smaller quantifier rank.

For every $q \in \mathbb{N}$ denote by FO_q the set of sentences of quantifier rank at most q. There are only finitely pairwise nonequivalent sentences in FO_q (say in the language for graphs). Thus we see that there is no sequence $(\varphi_\ell)_{\ell \in \mathbb{N}}$ of bounded quantifier rank such that for every $k \in \mathbb{N}$ the sentence φ_k is equivalent to ψ_k, that is, defines the kth slice of p-Vertex-Cover. Nevertheless our first main result reads as follows:

▶ **Theorem 1.** *p-Vertex-Cover is slicewise definable in FO_{17}.*

So we have to explain what we mean by slicewise definable in Theorem 1 (the precise definition will be given in Definition 6).

It is well known that first-order logic FO captures the complexity class uniform AC^0. This result crucially relies on the assumption that the input graphs (or more generally, the input structures) are equipped with built-in addition and multiplication. Our notion of slicewise definability assumes that the graphs have built-in addition and multiplication and furthermore, constants for an initial segment of natural numbers of a length depending only on the parameter.

The vertex cover problem is a special case of the *hitting set problem* on hypergraphs of bounded hyperedge size. For every $d \in \mathbb{N}$ a d-hypergraph is a hypergraph with hyperedges of size at most d. Then, the *parameterized d-hitting set problem* p-d-Hitting-Set asks whether an input d-hypergraph $\mathcal{G}$ contains a set of k vertices that intersects with every hyperedge in $\mathcal{G}$. Thus p-Vertex-Cover is basically the parameterized 2-hitting set problem. Extending Theorem 1 we prove that p-d-Hitting-Set is slicewise definable in FO_q, where $q = O(d^2)$. The problem p-d-Hitting-Set can be Fagin-defined [8] by a FO-formula with a second-order variable which does not occur in the scope of an existential quantifier or negation symbol. We show that all problems Fagin-definable in this form are slicewise definable in some FO_q.

What is the complexity of the class of parameterized problems that are slicewise definable in FO with bounded quantifier rank? We prove that it coincides with para-FO [6], the class of problems FO-definable after a precomputation on the parameter. Thus we obtain a descriptive characterization of the class para-FO, or equivalently of the parameterize circuit complexity class para-AC^0 [7, 3, 6]. The equivalence between para-FO and para-AC^0 is an easy consequence of the equivalence between FO and the classical circuit complexity class uniform AC^0 mentioned above.

The main technical tool for proving Theorem 1 and the subsequent results, the *color coding* method [1], makes essential use of arithmetic. The counting quantifier $\exists^{\geq k} x$ in $\exists^{\geq k} x \varphi(x)$ is

an abbreviation for the FO-sentence

$$\exists x_1 \cdots \exists x_k \Big(\bigwedge_{1 \le i < j \le k} x_i \neq x_j \wedge \bigwedge_{1 \le i \le k} \varphi(x_i) \Big)$$

expressing that there are at least k many elements satisfying φ. Clearly, $\mathrm{qr}(\exists^{\geq k} x \varphi(x)) = k + \mathrm{qr}(\varphi(x))$, so the sequence $(\exists^{\geq k} x \varphi(x))_{k \in \mathbb{N}}$ has unbounded quantifier rank. Using the color coding method we get a sequence $(\chi^k)_{k \in \mathbb{N}}$ of bounded quantifier rank such that each χ^k is equivalent to $\exists^{\geq k} x \varphi(x)$. Note that the sentence in (1) is not of the form $\exists^{\geq k} x \varphi(x)$. We exploit the idea underlying Buss' kernilization in order to get an FO-sentence expressing the existence of a vertex cover of size k in terms of counting quantifiers. Altogether, Theorem 1 (and its proof) exhibit the power of addition and multiplication, although on the face of it, the vertex cover problem has nothing to do with arithmetic operations.

In finite model theory there is consensus that inexpressibility results for FO and for fragments of FO are very hard to obtain in the presence of addition and multiplication. To get such a result we exploit the equivalence between FO and uniform AC^0, more precisely, we analyze the connection between the quantifier rank of a sentence φ and the depth of the corresponding AC^0 circuits. Together with a theorem [11, 15] on a version of Sipser functions we show that the hierarchy $(\mathrm{FO}_q)_{q \in \mathbb{N}}$ is strict:

▶ **Theorem 2.** *Let $q \in \mathbb{N}$. Then there is a parameterized problem slicewise definable in FO_{q+1} but not in FO_q.*

1.1 Organization of the paper

In Section 2 we prove Theorem 1, and then extend it to the hitting set problem in Section 3. We give a natural class of Fagin-definable problems that are slicewise definable in FO with bounded quantifier rank in Section 4. We prove the hierarchy theorem, i.e., Theorem 2, in Section 6. In the final section we conclude with some open problems. Due to space limitations we defer some proofs to the full version of the paper.

1.2 Some logic preliminaries

A *vocabulary* τ is a finite set of relation symbols. Each relation symbol has an *arity*. A *structure* $\mathcal{A}$ of vocabulary τ, or *τ-structure*, consists of a nonempty set A called the *universe* of $\mathcal{A}$, and of an interpretation $R^{\mathcal{A}} \subseteq A^r$ of each r-ary relation symbol $R \in \tau$. In this paper all structures have a finite universe. Occasionally we allow the use of constants: For a vocabulary τ we consider $(\tau \cup \{c_1, \ldots, c_s\})$-structures $\mathcal{A}$. Then $c_1^{\mathcal{A}}, \ldots, c_s^{\mathcal{A}}$, the interpretations of the constants $c_1, \ldots, c_s$, are elements of $\mathcal{A}$. However the letters τ, τ', ... will always denote *relational* vocabularies (without constants). If τ contains a binary relation symbol $<$ and in the structure $\mathcal{A}$ the relation $<^{\mathcal{A}}$ is an order of the universe, then $\mathcal{A}$ is an *ordered structure*.

Let τ be a vocabulary and C a set of constants. Formulas φ of first-order logic of vocabulary $\tau \cup C$ are built up from atomic formulas $t_1 = t_2$ and $Rt_1 \ldots t_r$ where $t_1, t_2, \ldots, t_r$ are either variables or constants in C, and where $R \in \tau$ is of arity r, using the boolean connectives and existential and universal quantification. A formula φ is a *sentence* if it has no free variables. The quantifier rank of φ is defined inductively as:

$$\mathrm{qr}(\varphi) := 0 \text{ if } \varphi \text{ is atomic} \qquad \mathrm{qr}(\varphi_1 \wedge \varphi_2) = \mathrm{qr}(\varphi_1 \vee \varphi_2) := \max\{\mathrm{qr}(\varphi_1), \mathrm{qr}(\varphi_2)\}$$
$$\mathrm{qr}(\neg\varphi) := \mathrm{qr}(\varphi) \qquad \mathrm{qr}(\exists x \varphi) = \mathrm{qr}(\forall x \varphi) = 1 + \mathrm{qr}(\varphi).$$

2 Slicewise-definability in FO_q and the vertex cover problem

In this section we prove Theorem 1, i.e., p-Vertex-Cover is slicewise definable in FO_{17}. Our main tool is Theorem 4. It shows how we can express that there are k elements having a first-order property by a number of quantifiers independent of k. We give further applications of this tool in this and the next section.

For $n \in \mathbb{N}$ let $[n] := \{0, 1, \ldots, n-1\}$. Denote by $<^{[n]}$ the natural order on $[n]$. Clearly, if $\mathcal{A}$ is any ordered structure, then $(A, <^{\mathcal{A}})$ is isomorphic to $\big([|A|], <^{[|A|]}\big)$ and the isomorphism is unique. For ternary relation symbols $+$ and $\times$ we consider the ternary relations $+^{[n]}$ and $\times^{[n]}$ on $[n]$ that are the relations of addition and multiplication of $\mathbb{N}$ restricted to $[n]$. That is, $+^{[n]} := \{(a,b,c) \mid a,b,c \in [n] \text{ with } c = a+b\}$; $\times^{[n]} := \{(a,b,c) \mid a,b,c \in [n] \text{ with } c = a \cdot b\}$. Finally, for every $m \in \mathbb{N}$ let $C(m) := \{\overline{\ell} \mid \ell < m\}$ be a set of constants and set

$$\overline{\ell}^{\,[n]} := \ell, \quad \text{if } \ell < n \qquad \text{and} \qquad \overline{\ell}^{\,[n]} := n-1, \quad \text{if } \ell \geq n.$$

The letters τ, τ', ... will always denote *relational* vocabularies (without constants). Assume τ contains $<$, $+$, and $\times$. A $(\tau \cup C(m))$-structure $\mathcal{A}$ *has built-in* $<, +, \times, C(m)$ if its $\{<, +, \times, C(m)\}$-reduct is isomorphic to $\big([n], <^{[n]}, +^{[n]}, \times^{[n]}, (\overline{\ell}^{\,[n]})_{\ell<m}\big)$.

If $m = 0$, we briefly say that $\mathcal{A}$ *has built-in addition and multiplication*. We denote by $\mathrm{ARITHM}[\tau]$ the class of τ-structures with built-in addition and multiplication. If $\mathcal{A} \in \mathrm{ARITHM}[\tau]$ and $m \in \mathbb{N}$, we denote by $\mathcal{A}_{C(m)}$ its unique expansion to a $(\tau \cup C(m))$-structure with built-in $<, +, \times, C(m)$.

In the proof of Theorem 4 we use Lemma 3, the color coding technique of Alon et al. [1] essentially in the form presented in [10, Claim 1 on page 349]. It will enable us to find in every sufficiently large structure with built-in arithmetic for each subset X of cardinality k an FO-definable function that one-to-one maps X into the initial segment of length k^2.

▶ **Lemma 3.** *There is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, all $k \leq n$ and for every k-element subset X of $[n]$, there exists a prime $p < k^2 \cdot \log_2 n$ and a $q < p$ such that the function $h_{p,q} : [n] \to \{0, \ldots, k^2 - 1\}$ given by $h_{p,q}(m) := (q \cdot m \mod p) \mod k^2$ is injective on X.*

As already mentioned the following result allows to express the existence of k elements satisfying a first-order property by a number of quantifiers independent of k.

▶ **Theorem 4.** *Let τ be a vocabulary containing $<$, $+$, $\times$. Then there is an algorithm that assigns to every $k \in \mathbb{N}$ and every $\mathrm{FO}[\tau]$-formula $\varphi(\bar{x}, y)$ an $\mathrm{FO}\big[\tau \cup C(k^2+1)\big]$-formula $\chi^k_\varphi(\bar{x})$ such that for every $\mathcal{A} \in \mathrm{ARITHM}[\tau]$ with $k^2 \leq |A|/\log |A|$ and $|A| \geq n_0$ and $\bar{u} \in A$,*

$$\begin{aligned} &\mathcal{A}_{C(k^2)} \models \chi^k_\varphi(\bar{u}) \iff \\ &\quad \text{there are pairwise distinct } v_0, \ldots, v_{k-1} \in A \text{ with } \mathcal{A} \models \varphi(\bar{u}, v_i) \text{ for every } i \in [k]. \end{aligned} \tag{2}$$

By Lemma 3 we can take as $\chi^k_\varphi(\bar{x})$ a formula expressing

$$\exists p \exists q \Big(\bigvee_{0 \leq i_0 < \cdots < i_{k-1} < k^2} \bigwedge_{j \in [k]} \exists y \big(\text{“}h_{p,q}(y) = i_j \text{”} \wedge \varphi(\bar{x}, y) \big) \Big).$$

Furthermore, $\mathrm{qr}\big(\chi^k_\varphi(\bar{x})\big) = \max\big\{12, \mathrm{qr}\big(\varphi(\bar{x}, y)\big) + 3\big\}$.

Note that the conditions “$k^2 \leq |A|/\log |A|$ and $|A| \geq n_0$” on $|A|$ are fulfilled if $|A| \geq \max\{2^{k^2}, n_0\}$, so we have a lower bound of $|A|$ in terms of k (here n_0 is a natural number according to Lemma 3).

Proof of Theorem 4. Let $\mathcal{A}$ be as above, set $n := |A|$, and w.l.o.g. assume that $A := [n]$. In order to make formulas more readable, we introduce some abbreviations. Clearly, $x = (y \bmod z)$ is an abbreviation for $\exists u(y = u \times z + x \wedge x < z)$, more precisely, as $+$ and $\times$ are relation symbols, an abbreviation for $\exists u \exists u'(u' = u \times z \wedge y = u' + x \wedge x < z)$. Now let

$$\chi^k_\varphi(\bar{x}) := \exists p \exists q \Big(\bigvee_{0 \le i_1 < \cdots < i_{k-1} < k^2} \bigwedge_{j \in [k]} \exists y (\text{“}h_{p,q}(y) = i_j\text{”} \wedge \varphi(\bar{x}, y)) \Big),$$

where

$$\text{“}h_{p,q}(y) = i_j\text{”} := (q \times (u \mod p) \mod p) \mod \overline{k^2} = \overline{i_j}.$$

We replaced $(q \times u \bmod p)$ by $(q \times (u \bmod p) \bmod p)$, since $q \times u$ might exceed $|A|$. To count the quantifier rank note that “$h_{p,q}(y)$” $= \overline{i_j}$ means

$$\exists v \exists v' \exists \alpha \left(v' = v \times \overline{k^2} \wedge \alpha = v' + \overline{i_j} \wedge \overline{i_j} < \overline{k^2} \right),$$

where the intended meaning of α is $(q \times (u \bmod p) \bmod p)$. So α is the unique element satisfying

$$\exists w \exists w' \exists \beta (w' = w \times p \wedge \beta = w' + \alpha \wedge \alpha < p).$$

Here the intended meaning of β is $q \times (u \bmod p)$. Thus β is the unique element satisfying

$$\exists \gamma (\beta = q \times \gamma \wedge \text{“}\gamma = u \mod p\text{”}).$$

So we can replace “$\gamma = u \bmod p$” by

$$\exists z \exists z' (z' = z \times p \wedge u = z' + \gamma \wedge \gamma < p).$$

Thus, $\mathrm{qr}(\text{“}h_{p,q}(y) = i_j\text{”}) = 9$ and hence, $\mathrm{qr}(\chi^k_\varphi(\bar{x})) = \max\{12, \mathrm{qr}(\varphi(\bar{x}, y)) + 3\}$. ◀

We use the previous result to show that two parameterized problems are slicewise definable in FO_q for some q, one is an easy application, the other the more intricate p-Vertex-Cover. First we give the precise definitions of parameterized problem in our context and of slicewise definability.

▶ **Definition 5.** A parameterized problems is a subclass Q of $\mathrm{ARITHM}[\tau] \times \mathbb{N}$ for some vocabulary τ, where for each $k \in \mathbb{N}$ the class $Q_k := \{\mathcal{A} \in \mathrm{ARITHM}[\tau] \mid (\mathcal{A}, k) \in Q\}$ is closed under isomorphism. The class Q_k is the *kth slice of* Q.

Every pair $(\mathcal{A}, k) \in \mathrm{ARITHM}[\tau] \times \mathbb{N}$ is an *instance* of Q, $\mathcal{A}$ its *input* and k its *parameter*.

▶ **Definition 6.** Q is *slicewise definable in* FO *with bounded quantifier rank*, briefly $Q \in \mathrm{XFO}_{\mathrm{qr}}$, if there is a $q \in \mathbb{N}$ and computable functions $h : \mathbb{N} \to \mathbb{N}$ and $f : \mathbb{N} \to \mathrm{FO}_q[\tau \cup C(h(k))]$ such that for all $(\mathcal{A}, k) \in \mathrm{ARITHM}[\tau] \times \mathbb{N}$,

$$(\mathcal{A}, k) \in Q \iff \mathcal{A}_{C(h(k))} \models f(k).$$

That is, if $m_k := h(k)$ and $\varphi_k := f(k)$, then

$$(\mathcal{A}, k) \in Q \iff \mathcal{A}_{C(m_k)} \models \varphi_k.$$

We then say that Q *is slicewise definable in* FO_q and write $Q \in \mathrm{XFO}_q$.

Using the constants in $C(m)$ we can characterize arithmetical structures with less that m elements by a quantifier free sentence, more precisely:

▶ **Lemma 7.** *Assume that* $\mathcal{A} \in \mathrm{ARITHM}[\tau]$ *and that* $|A| < m$*. Then there is a quantifier free* $\mathrm{FO}[\tau \cup C(m)]$*-sentence* $\varphi_{\mathcal{A}_{C(m)}}$ *(that is,* $\varphi_{\mathcal{A}_{C(m)}} \in \mathrm{FO}_0[\tau \cup C(m)]$*) such that for all structures* $\mathcal{B} \in \mathrm{ARITHM}[\tau]$ *we have*

$$\mathcal{B}_{C(m)} \models \varphi_{\mathcal{A}_{C(m)}} \iff \mathcal{A} \cong \mathcal{B}.$$

Using this lemma we get the following simple but useful observation.

▶ **Proposition 8.** *Let* $Q \in \mathrm{ARITHM}[\tau] \times \mathbb{N}$ *be a decidable parameterized problem and* $q \in \mathbb{N}$*. Assume that* Q *is* eventually slicewise definable in FO_q*, that is, there are computable functions* $k \mapsto m_k$ *with* $m_k \in \mathbb{N}$ *and* $k \mapsto \varphi_k$ *with* $\varphi_k \in \mathrm{FO}_q\big[\tau \cup C(m_k)\big]$ *and a computable and increasing function* $g : \mathbb{N} \to \mathbb{N}$ *such that for all* $(\mathcal{A}, k) \in \mathrm{ARITHM}[\tau] \times \mathbb{N}$ *with* $|A| \geq g(k)$*,*

$$(\mathcal{A}, k) \in Q \iff \mathcal{A}_{C(m_k)} \models \varphi_k.$$

Then Q *is slicewise definable in* FO_q*.*

We now turn to our first application of Theorem 4.

▶ **Theorem 9.** *The parameterized problem*

p-deg-Independent-Set	
Input:	A graph $\mathcal{G}$.
Parameter:	$k \in \mathbb{N}$.
Question:	Is $k \geq \deg(\mathcal{G})$ and does $\mathcal{G}$ have an independent set of $k - \deg(\mathcal{G})$ elements?

is slicewise definable in FO_{13}*.*

Let $\tau_{\text{Graph}} := \{E, +, <, \times\}$ with binary E. More formally, in our context we mean by *p-deg*-Independent-Set the following class:

$$\Big\{(\mathcal{G}, k) \in \mathrm{ARITHM}[\tau_{\text{Graph}}] \times \mathbb{N} \;\Big|\; k \geq \deg(\mathcal{G}) \text{ and } \\ \text{(the } \{E\}\text{-reduct of) } \mathcal{G} \text{ has an independent set of size } \ell := k - \deg(\mathcal{G})\Big\}.^1$$

Proof. An easy induction on $\ell := k - \deg(\mathcal{G})$ shows that every graph $\mathcal{G}$ with at least $(\deg(\mathcal{G}) + 1) \cdot \ell$ vertices has an independent set of size ℓ. Hence, for $(\mathcal{G}, k) \in \mathrm{ARITHM}[\tau]$, where the graph $\mathcal{G}$ has at least $(k+1) \cdot k$ vertices, we have

$$(\mathcal{G}, k) \in \mathrm{ARITHM}[\tau_{\text{Graph}}] \in \textit{p-deg}\text{-Independent-Set} \iff k \geq \deg(\mathcal{G}). \tag{3}$$

We use this fact to prove that *p-deg*-Independent-Set is eventually slicewise definable in FO_{13}, which yields our claim by Proposition 8.

Let $d \in \mathbb{N}$ and $\varphi := Euy$. Then, by Theorem 4, we have for every graph $\mathcal{G}$ with at least $h(k)$ vertices for some computable $h : \mathbb{N} \to \mathbb{N}$ and every vertex u of $\mathcal{G}$,

$$\mathcal{G} \models \chi^d_\varphi(u) \iff \text{the degree of } u \text{ in } \mathcal{G} \text{ is} \geq d.$$

[1] In the following we will present parameterized graph problems in the more liberal form as given by the box above.

Thus, for $k \in \mathbb{N}$ and every graph $\mathcal{G} \in \text{ARITHM}[\tau_{\text{GRAPH}}]$ with at least $\max\{h(k), (k+1)\cdot k\}$ vertices, by (3),

$$(\mathcal{G}, k) \in p\text{-}deg\text{-}\textsc{Independent-Set} \iff \mathcal{G} \models \neg\exists u \chi_\varphi^{k+1}(u).$$

As $\text{qr}(\varphi) = 0$, Theorem 4 and the previous equivalence show that p-*deg*-INDEPENDENT-SET is eventually in XFO_{13} (and hence in XFO_{13} by Proposition 8). ◀

Now we are ready to show the slicewise definability of p-VERTEX-COVER in FO_{17}.

Proof of Theorem 1. Recall the main ingredient of Buss' kernelization for an instance $(\mathcal{G}, k)$ of the vertex cover problem.

1. If a vertex v has degree $\geq k+1$ in $\mathcal{G}$, then v must be in every vertex cover of size k. We remove all v of degree $\geq k+1$ in $\mathcal{G}$, say ℓ many, and decrease k to $k' := k - \ell$.
2. Remove all isolated vertices.
3. Let $\mathcal{G}'$ be the resulting induced graph. If $k' < 0$ or $\mathcal{G}'$ has $> k' \cdot (k+1)$ vertices, then $(\mathcal{G}', k')$, and hence also $(\mathcal{G}, k)$, is a NO instance of p-VERTEX-COVER.

Again let $\varphi(x, y) := Exy$. Then, by Theorem 4, for every instance $(\mathcal{G}, k)$ of p-VERTEX-COVER, where the vertex set G of $\mathcal{G}$ is sufficiently large compared with k and every vertex $v \in G$,

$$\mathcal{G} \models \chi_\varphi^{k+1}(v) \iff v \text{ has degree } \geq k+1.$$

Therefore, applying again Theorem 4 we get for $\ell \in \mathbb{N}$,

$$\mathcal{G} \models \left(\chi^{\ell}_{\chi_\varphi^{k+1}} \wedge \neg\chi^{\ell+1}_{\chi_\varphi^{k+1}}\right) \iff \mathcal{G} \text{ has exactly } \ell \text{ vertices of degree } \geq k+1.$$

For every vertex v of $\mathcal{G}$ we have

$$\mathcal{G} \models \mathit{uni}(v) \iff v \text{ is a vertex of } \mathcal{G}',$$

where

$$\mathit{uni}(x) := \left(\neg\chi_\varphi^{k+1}(x) \wedge \neg\forall y\big(Exy \to \chi_\varphi^{k+1}(y)\big)\right).$$

Then,

$$\begin{aligned}
&(\mathcal{G},k) \in p\text{-}\textsc{Vertex-Cover}\\
&\quad\iff \text{for some } \ell \text{ with } 0 \leq \ell \leq k, \mathcal{G} \text{ has exactly } \ell \text{ vertices of degree } \geq k+1 \text{ and}\\
&\qquad\qquad \text{there is a } j \leq (k-\ell)\cdot(k+1) \text{ such that } \mathcal{G}' \text{ has } j \text{ vertices and}\\
&\qquad\qquad (\mathcal{G}', k-\ell) \text{ is a YES instance of } p\text{-}\textsc{Vertex-Cover}\\
&\quad\iff \mathcal{G} \models \bigvee_{0\leq\ell\leq k}\Big(\chi^{\ell}_{\chi_\varphi^{k+1}} \wedge \neg\chi^{\ell+1}_{\chi_\varphi^{k+1}} \wedge \bigvee_{0\leq j\leq(k-\ell)\cdot(k+1)} (\chi^{j}_{uni} \wedge \neg\chi^{j+1}_{uni} \wedge \rho_j)\Big). \qquad (4)
\end{aligned}$$

Here the formula ρ_j, a formula expressing (in $\mathcal{G}$ with an $\mathcal{G}'$ with exactly j vertices) that $\mathcal{G}'$ has a vertex cover of size $k-\ell$, still has to be defined. We do that by saying that $\mathcal{G}'$ is isomorphic to one of the graphs with j vertices that has a vertex cover of size $k-\ell$. For this we have to be able to define an order of $\mathcal{G}'$ by a formula of quantifier rank bounded by a constant number independent of k. Again this is done (using built-in arithmetics) with the color coding method: We find p and q, and $0 \leq i_0 < \cdots < i_{j-1} < j^2$ with

$$h_{p,q}(G') = \{i_0, \ldots, i_{j-1}\}.$$

Then, we can speak of the first, the second ,..., vertex in $\mathcal{G}'$. Thus as ρ_j we can take the sentence

$$\exists p \exists q \Big(\bigvee_{0 \leq i_0 < \ldots < i_{j-1} < j^2} \Big(\bigwedge_{s \in [j]} \exists y \big(\text{“}h_{p,q}(y) = i_s\text{”} \wedge \mathit{uni}(y) \big) \wedge \bigvee_{\substack{(\mathcal{H}, k-\ell) \in p\text{-Vertex-Cover} \\ H = [j]}} \rho'_{\mathcal{H}} \Big) \Big),$$

where

$$\rho'_{\mathcal{H}} := \bigwedge_{\substack{s,t \in [j] \\ E^{\mathcal{H}} st}} \exists y \exists z \big(\mathit{uni}(y) \wedge \mathit{uni}(z) \wedge \text{“}h_{p,q}(y) = i_s\text{”} \wedge \text{“}h_{p,q}(z) = i_t\text{”} \wedge Eyz \big)$$
$$\wedge \bigwedge_{\substack{s,t \in [j] \\ \neg E^{\mathcal{H}} st}} \exists y \exists z \big(\mathit{uni}(y) \wedge \mathit{uni}(z) \wedge \text{“}h_{p,q}(y) = i_s\text{”} \wedge \text{“}h_{p,q}(z) = i_t\text{”} \wedge \neg Eyz \big).$$

As $\mathrm{qr}(\chi_\varphi^{k+1}) \leq 12$, we have $\mathrm{qr}(\mathit{uni}(x)) \leq 13$. Thus, $\mathrm{qr}(\chi^j_{\mathit{uni}}) \leq 16$ and $\mathrm{qr}(\rho_j) \leq 17$. As the remaining formulas in (4) have at most quantifier rank 16, we get p-Vertex-Cover $\in \mathrm{XFO}_{17}$. ◀

3 The hitting set problems with bounded hyperedge size

We consider the parameterized problem

p-d-Hitting-Set	
Input:	A hypergraph $\mathcal{G}$ with edges of size at most d.
Parameter:	$k \in \mathbb{N}$.
Question:	Does $\mathcal{G}$ have a hitting set of size k?

Here a *hypergraph* $\mathcal{G}$ is a pair (V, E), where V is a set, the set of *vertices of* $\mathcal{G}$, and every element of E is a *hyperedge*, that is, a nonempty subset of V. A *hitting set* in $\mathcal{G}$ is a set H that intersects each hyperedge (that is, $H \cap e \neq \emptyset$ for all $e \in E$).

The goal of this section is to show:

▶ **Theorem 10.** *Let $d \geq 1$. Then p-d-Hitting-Set is slicewise definable in* FO *with bounded quantifier rank; more precisely, p-d-Hitting-Set $\in \mathrm{XFO}_q$ with $q = O(d^2)$.*

The following lemma can be viewed as a generalization of part of Buss' kernelization algorithm for p-Vertex-Cover to p-d-Hitting-Set. The case for p-3-Hitting-Set was first shown in [13].

▶ **Lemma 11.** *Let $(\mathcal{G}, k)$ with $\mathcal{G} = (V, E)$ be an instance of p-d-Hitting-Set. Let $1 < \ell \leq d$ and assume that every ℓ-set (i.e., set with exactly ℓ elements) of vertices has at most $k^{d-\ell}$ extensions in E.*

If $v_1, \ldots, v_{\ell-1}$ are pairwise distinct vertices such that there is a hitting set H of size $\leq k$ that contains none of these vertices, then $\{v_1, \ldots, v_{\ell-1}\}$ has at most $k^{d-(\ell-1)}$ extensions in E.

Proof. Every hyperedge that extends $\{v_1, \ldots, v_{\ell-1}\}$ must contain a vertex u of the hitting set H. By the assumptions, u is distinct from the v_i's and therefore, the set $\{v_1, \ldots, v_{\ell-1}, u\}$ has at most $k^{d-\ell}$ extensions in E. As $|H| \leq k$, we see that there are at most $k \cdot k^{d-\ell}$ $\big(= k^{d-(\ell-1)}\big)$ extensions in E. ◀

Let $(\mathcal{G}, k)$ and $1 < \ell \leq d$ satisfy the hypotheses of the lemma, that is, $(\mathcal{G}, k)$ with $\mathcal{G} = (V, E)$ is an instance of p-d-Hitting-Set and every ℓ-set has at most $k^{d-\ell}$ extensions in E. For every pairwise distinct vertices $v_1, \ldots, v_{\ell-1}$ such that $\{v_1, \ldots, v_{\ell-1}\}$ has more than $k^{d-(\ell-1)}$ extensions in E, we delete from E all hyperedges extending $\{v_1, \ldots, v_{\ell-1}\}$ and add the hyperedge $\{v_1, \ldots, v_{\ell-1}\}$. Let $\mathcal{G}^\ell = (V, E^\ell)$ be the resulting hypergraph. Then:

(a) For every pairwise distinct vertices $v_1, \ldots, v_{\ell-1}$ there are at most $k^{d-(\ell-1)}$ hyperedges in E^ℓ extending $\{v_1, \ldots, v_{\ell-1}\}$.

(b) If H is a subset of V and $|H| \leq k$, then

$$H \text{ is a hitting set of } \mathcal{G} \iff H \text{ is a hitting set of } \mathcal{G}^\ell,$$

in particular,

$$(\mathcal{G}, k) \in p\text{-}d\text{-Hitting-Set} \iff (\mathcal{G}^\ell, k) \in p\text{-}d\text{-Hitting-Set}.$$

Let $(\mathcal{G}, k)$ be an instance of p-d-Hitting-Set. For $\ell := d$ the hypothesis of Lemma 11 is fulfilled: Every d-set of vertices has at most one extension in E, namely at most, itself. Hence, applying the above procedure for $\ell = d$ we get the hypergraph $\mathcal{G}^\ell$, which satisfies the hypotheses of Lemma 11 for $\ell := d - 1$. So we get, again by the above procedure the hypergraph $(\mathcal{G}^\ell)^{\ell-1}$, which we denote by $\mathcal{G}^{\ell,\ell-1}$. Following this way, we finally obtain the hypergraph $\mathcal{G}^{\ell,\ell-1,\ldots,2}$, which we denote by $\mathcal{G}'$. Note that $\mathcal{G}' = (V, E')$ for some E'. From (a) and (b) we get (a') and (b').

(a') For every vertex v there are at most k^{d-1} hyperedges in E' containing v.

(b') If H is a subset of V and $|H| \leq k$, then

$$H \text{ is a hitting set of } \mathcal{G} \iff H \text{ is a hitting set of } \mathcal{G}',$$

Moreover,

(c') If $(\mathcal{G}, k) \in p$-d-Hitting-Set, then $|E'| \leq k^d$ and $|V'| \leq d \cdot k^d$, where $V' := \{v \in V \mid \text{there is an } e \in E' \text{ with } v \text{ in } e\}$ is the set of non-isolated vertices of $\mathcal{G}'$.

In fact, let H be a hitting set with $|H| = k$ of $\mathcal{G}$ and hence, by (b') of $\mathcal{G}'$. As every hyperedge must contain a vertex of H, we get $|E'| \leq k^d$ from (a'). As every hyperedge $e \in E'$ contains at most d vertices, we have $|V'| \leq d \cdot k^d$.

We fix k and look at the kth slice of p-d-Hitting-Set. In the proof of Theorem 10 which will be presented in the full paper we will see that for hypergraphs $\mathcal{G}$ sufficiently large compared with k we can FO-define $\mathcal{G}'$ in $\mathcal{G}$. By (b') and (c'), we know that $(\mathcal{G}, k) \in p$-d-Hitting-Set implies $|E'| \leq k^d$. By Theorem 4, we can express $|E'| \leq k^d$ in first-order logic with a bounded number of quantifiers if we add built-in addition and multiplication. Essentially this shows that p-d-Hitting-Set is eventually slicewise definable in FO with bounded quantifier rank and thus, p-d-Hitting-Set $\in \mathrm{XFO}_{\mathrm{qr}}$ (by Proposition 8).

A part of an FO-interpretation I is an FO-formula $\varphi^I_{uni}(x_1, \ldots, x_s)$ defining the universe of the defined structure, that is: if I is an interpretation of σ-structures in a class $\mathbf{K}$ of τ-structures, then for every structure $\mathcal{A} \in \mathbf{K}$ the set

$$(\varphi^I_{uni})^{\mathcal{A}} := \{(a_1, \ldots, a_s) \in A^s \mid \mathcal{A} \models \varphi(a_1, \ldots, a_s)\}$$

is the universe of the σ-structure $I(\mathcal{A})$ defined by I in $\mathcal{A}$.

Assume that σ does not contain the relation symbols $<, +, \times$, but that the structures in $\mathbf{K}$ are structures with built-in addition and multiplication, i.e., $\mathbf{K} \subseteq \mathrm{ARITHM}[\tau]$. In general, we can not extend the interpretation I to an FO-interpretation J such that

$$J(\mathcal{A}) = \left(I(\mathcal{A}), <^{J(\mathcal{A})}, +^{J(\mathcal{A})}, \times^{J(\mathcal{A})}\right)$$

has built-in addition and multiplication (that is, so that $J(\mathcal{A})$ is $I(\mathcal{A})$ together with an order and the corresponding addition and multiplication).

For example, for $\tau = \{P, <, +, \times\}$ with unary P let $\mathbf{K}$ be the class of τ-structures $\mathcal{A}$ with $P^{\mathcal{A}} \neq \emptyset$. Let σ be the empty vocabulary and consider the interpretation I yielding in $\mathcal{A}$ the σ-structure with universe $P^{\mathcal{A}}$ (take $\varphi^I_{uni}(x) := Px$). If we could extend I to an interpretation J such that $J(\mathcal{A}) := (P^{\mathcal{A}}, <^{\mathcal{A}}, +^{\mathcal{A}}, \times^{\mathcal{A}})$ has built-in addition and multiplication, then we could express in $J(\mathcal{A})$, and thus in $\mathcal{A}$, that "$P^{\mathcal{A}}$ is even," i.e., the parity problem, which is well known to be impossible.

The next result (proven in [4, Lemma 10.5], see also [12, Exercise 1.33]) shows that the situation is different if for $\varphi^I_{uni}(x_1, \ldots, x_s)$ we have $(\varphi^I_{uni})^{\mathcal{A}} = A^s$.

▶ **Proposition 12.** *Let τ contain $<, +, \times$ and assume that none of these symbols is in σ. Let $\mathbf{K} \subseteq$ ARITHM$[\tau]$ and let I be an* FO*-interpretation of σ-structures in the structures in $\mathbf{K}$ with $\varphi^I_{uni} = \varphi^I_{uni}(x_1, \ldots, x_s)$. If for all $\mathcal{A} \in \mathbf{K}$, $(\varphi^I_{uni})^{\mathcal{A}} = A^s$, then I can be extended to an* FO*-interpretation of $\sigma \cup \{<, +, \times\}$ such that $J(\mathcal{A}) = (I(\mathcal{A}), <^{J(\mathcal{A})}, +^{J(\mathcal{A})}, \times^{J(\mathcal{A})})$ has built-in addition and multiplication for all $\mathcal{A} \in \mathbf{K}$.*

4 Fagin definability

Let $\varphi(X)$ be an FO$[\tau]$-formula which for a, say r-ary, second-order variable X may contain atomic formulas of the form $Xx_1 \ldots x_r$. Then the *parameterized problem* $\mathrm{FD}_{\varphi(X)}$ *Fagin-defined by* $\varphi(X)$ is the problem

$\mathrm{FD}_{\varphi(X)}$	
Input:	A τ-structure $\mathcal{A}$.
Parameter:	$k \in \mathbb{N}$.
Question:	Decide whether there is an $S \subseteq A^r$ with $\lvert S\rvert = k$ and $\mathcal{A} \models \varphi(S)$.

The following metatheorem improves [9, Theorem 4.4].

▶ **Theorem 13.** *Let $\varphi(X)$ be an* FO$[\tau]$*-formula without first-order variables occurring free and in which X does not occur in the scope of an existential quantifier or negation symbol. Then $\mathrm{FD}_{\varphi(X)} \in \mathrm{XFO}_{\mathrm{qr}}$ that is, $\mathrm{FD}_{\varphi(X)}$ is slicewise definable with bounded quantifier rank.*

We view a hypergraph $\mathcal{G} := (V, E)$ as an $\{E_0, \epsilon\}$-structure $(V \cup E, E_0^{\mathcal{G}}, \epsilon^{\mathcal{G}})$, where E_0 is a unary relation symbol and ϵ is a binary relation symbol and

$$E_0^{\mathcal{G}} := E \quad \text{and} \quad \epsilon^{\mathcal{G}} := \big\{(v, e) \;\big|\; v \in V,\, e \in E \text{ and } v \in e\big\}.$$

Fix $d \in \mathbb{N}$. For $k \in \mathbb{N}$ we have (assuming $\lvert V\rvert \geq k$)

$$(\mathcal{G}, k) \in p\text{-}d\text{-Hitting-Set} \iff \text{for some } S \text{ with } \lvert S\rvert = k \text{ we have } (V \cup E, E_0^{\mathcal{G}}, \epsilon^{\mathcal{G}}) \models \varphi(S),$$

where $\varphi(x) := \forall e\Big(E_0 e \to \forall x_1 \ldots \forall x_d\big((\forall x(x \,\epsilon\, e \leftrightarrow \bigvee_{i=1}^{d} x_i = x) \to (Xx_1 \vee \ldots \vee Xx_d)\big)\Big)$. By Theorem 13 we know that $\mathrm{FD}_{\varphi(X)} \in \mathrm{XFO}_{\mathrm{qr}}$. Hence, p-d-Hitting-Set $\in \mathrm{XFO}_{\mathrm{qr}}$, so we get the result of the previous section. However here, to prove Theorem 13 we use the result of the previous section.

Proof of Theorem 13. For simplicity, let us assume that X is unary. Without loss of generality we can assume that

$$\phi(X) = \forall y_1 \ldots \forall y_\ell \bigwedge_{i=1}^{m} \bigvee_{j=1}^{p} \psi_{ij},$$

where each ψ_{ij} either is Xy_q for some $q \in \{1, \dots, \ell\}$, or a first-order formula with free variables in $\{y_1, \dots, y_\ell\}$ in which X does not occur.

Let $(\mathcal{A}, k)$ be an instance of $\mathrm{FD}_{\varphi(X)}$. We construct an instance $(\mathcal{G}(\mathcal{A}), k)$ of p-ℓ-Hitting-Set such that

$$(\mathcal{A}, k) \in \mathrm{FD}_{\varphi(X)} \iff (\mathcal{G}(\mathcal{A}), k) \in p\text{-}\ell\text{-Hitting-Set}. \tag{5}$$

As $(\mathcal{G}(\mathcal{A}), k)$ we take the hypergraph (V, E) with $V = A$ and where E contains the following hyperedges. Let $\bar{a} \in A^\ell$ and $i \in \{1, \dots, m\}$. If

$$\mathcal{A} \models \neg \bigvee_{\substack{j \in \{1,\dots,p\} \\ X \text{ does not occur in } \psi_{ij}}} \psi_{ij}(\bar{a}).$$

then E contains the hyperedge $\{a_{s_1}, \dots, a_{s_t}\}$ where $Xy_{s_1}, \dots, Xy_{s_t}$ are exactly the disjuncts of the form $Xy_\dots$ in $\bigvee_{j=1}^{p} \psi_{ij}$. If $t = 0$ (for some $\bar{a} \in A^\ell$), we take as $\mathcal{G}(\mathcal{A})$ a fixed hypergraph chosen in advance such that $(\mathcal{G}(\mathcal{A}), k)$ is a no instance of p-ℓ-Hitting-Set.

Since $\mathcal{G}(\mathcal{A})$ can be defined from $\mathcal{A}$ by an FO-interpretation and p-ℓ-Hitting-Set $\in \mathrm{XFO}_{\mathrm{qr}}$, we get $\mathrm{FD}_{\varphi(X)} \in \mathrm{XFO}_{\mathrm{qr}}$. ◀

Some parameterized problems can be shown to be in para-FO by a simple application of this theorem, e.g., for every $\ell \geq 1$, the problem p-$\mathrm{WSat}(\Gamma^+_{1,\ell})$, the restriction of p-Dominating-Set to graphs of degree ℓ, and the problem p-ℓ-Matrix-Domination, the restriction to matrices with at most ℓ ones in every row and column in p-Matrix-Domination.

5 para-AC^0 = XFO$_{\mathrm{qr}}$

The importance of the class $\mathrm{XFO}_{\mathrm{qr}}$ from the point of view of complexity theory stems from the fact that it coincides with the class para-AC^0, the class of parameterized problems that are in dlogtime-uniform AC^0 after a precomputation. As dlogtime-uniform AC^0 contains precisely the class of parameterized problems definable in first-order logic, the class para-AC^0 corresponds to the class para-FO of parameterized problems definable in first-order logic after a precomputation on the parameter (see [7, 6]). We deal here with the class para-FO and thus in this section aim to show para-FO = $\mathrm{XFO}_{\mathrm{qr}}$.

To define the class para-FO we need a notion of union of two arithmetical structures.

▶ **Definition 14.** Assume $\mathcal{A} \in \mathrm{ARITHM}[\tau]$ and $\mathcal{A}' \in \mathrm{ARITHM}[\tau']$ satisfy

$$A \cap A' = \emptyset \quad \text{and} \quad \tau \cap \tau' = \{<, +, \times\}.$$

Let U be a new unary relation symbol. We set $\tau \uplus \tau' := \tau \cup \tau' \cup \{U\}$. Then $\mathcal{A} \uplus \mathcal{A}'$ is the structure $\mathcal{B} \in \mathrm{ARITHM}(\tau \uplus \tau')$ with

- $B := A \cup A'$;
- $U^{\mathcal{B}} = A'$;
- $<^{\mathcal{B}} := <^{\mathcal{A}} \cup <^{\mathcal{A}'} \cup \{(a, a') \mid a \in A \text{ and } a' \in A'\}$, that is, the order $<^{\mathcal{B}}$ extends the orders $<^{\mathcal{A}}$ and $<^{\mathcal{A}'}$, and in $<^{\mathcal{B}}$ every element of A precedes every element of A';
- $R^{\mathcal{B}} := R^{\mathcal{A}}$ for $R \in \tau$ and $R^{\mathcal{B}} := R^{\mathcal{A}'}$ for $R \in \tau'$.

As we require $\mathcal{B} \in \mathrm{ARITHM}(\tau \uplus \tau')$ we do not need to define $+^{\mathcal{B}}$ and $\times^{\mathcal{B}}$ explicitly. If $A \cap A' \neq \emptyset$, then we pass to isomorphic structures with disjoint universes before defining $\mathcal{A} \uplus \mathcal{A}'$.

▶ **Definition 15.** Let $Q \subseteq \text{ARITHM}[\tau] \times \mathbb{N}$ be a parameterized problem. Q is *first-order definable after a precomputation*, in symbols $Q \in$ para-FO, if for some vocabulary τ' there is a computable function $pre : \mathbb{N} \to \text{ARITHM}[\tau']$, a *precomputation*, and a sentence $\varphi \in \text{FO}\big[\tau \uplus \tau'\big]$ such that for all $(\mathcal{A}, k) \in \text{ARITHM}[\tau] \times \mathbb{N}$,

$$(\mathcal{A}, k) \in Q \iff \mathcal{A} \uplus pre(k) \models \varphi.$$

The main result of this section reads as follows. It is the modeltheoretic analogue of the equivalence between (i) and (ii) of [6, Proposition 6].[2]

▶ **Theorem 16.** para-FO $= \text{XFO}_{\text{qr}}$.

Proof. We sketch a proof, details will be given in the full version of this paper. Assume that $Q \in$ para-FO. Hence, for some vocabulary τ' there is a computable function $pre : \mathbb{N} \to \text{ARITHM}[\tau']$ and a sentence $\varphi \in \text{FO}[\tau \uplus \tau']$ such that for all $(\mathcal{A}, k) \in \text{ARITHM}[\tau] \times \mathbb{N}$,

$$(\mathcal{A}, k) \in Q \iff \mathcal{A} \uplus pre(k) \models \varphi.$$

Clearly, then Q is decidable. Therefore, by Proposition 8, it suffices to show that for some $q \in \mathbb{N}$ the problem Q is eventually slicewise definable in FO_q, that is, that there are an increasing and computable function $g : \mathbb{N} \to \mathbb{N}$ and computable functions $k \mapsto m_k$ and $k \mapsto \psi_k \in \text{FO}_q[\tau \cup C(m_k)]$ such that for all $(\mathcal{A}, k) \in \text{ARITHM}[\tau] \times \mathbb{N}$ with $|A| \geq g(k)$,

$$\mathcal{A} \uplus pre(k) \models \varphi \iff \mathcal{A}_{C(m_k)} \models \psi_k. \tag{6}$$

The main idea: As the precomputation pre is computable, for $(\mathcal{A}, k) \in \text{ARITHM}[\tau] \times \mathbb{N}$ with sufficiently large $|A|$ compared with $|pre(k)|$, we can FO-define $pre(k)$ in $\mathcal{A}_{C(k+1)}$. Furthermore, from $\mathcal{A}$ and from this FO-defined $pre(k)$ in $\mathcal{A}_{C(k+1)}$ we get (an isomorphic copy of) $\mathcal{A} \uplus pre(k)$ in $\mathcal{A}_{C(k+1)}$ by an FO-interpretation. Summing up, we can FO-interpret $\mathcal{A} \uplus pre(k)$ in $\mathcal{A}_{C(k+1)}$. This FO-interpretation yields the desired ψ_k satisfying (6).

Now assume that $Q \in \text{FO}_{\text{qr}}$. Then there is a $q \in \mathbb{N}$ and computable functions $k \mapsto m_k$ with $m_k \in \mathbb{N}$ and $k \mapsto \varphi_k$ with $\varphi_k \in \text{FO}_q\big[\tau \cup C(m_k)\big]$ such that for all $(\mathcal{A}, k) \in \text{ARITHM}[\tau] \times \mathbb{N}$,

$$(\mathcal{A}, k) \in Q \iff \mathcal{A}_{C(m_k)} \models \varphi_k.$$

We have to find a precomputation $pre : \mathbb{N} \to \text{ARITHM}[\tau']$ and an $\text{FO}[\tau \uplus \tau']$-sentence φ such that for all $(\mathcal{A}, k) \in \text{ARITHM}[\tau] \times \mathbb{N}$,

$$\mathcal{A}_{C(m_k)} \models \varphi_k \iff \mathcal{A} \uplus pre(k) \models \varphi.$$

Essentially $pre(k)$ is the parse tree of φ_k and the sentence φ expresses that $\mathcal{A}_{C(m_k)}$ satisfies the sentence given by this parse tree, that is, the sentence φ_k. ◀

▶ **Corollary 17.** *For every $d \in \mathbb{N}$, p-d-*Hitting-Set *is in* para-FO *(and hence in* para-AC^0*).*

[2] Proposition 6 in [6] contains a third statement equivalent to (i) and (ii). The corresponding modeltheoretic analogue *decidable and eventually in* FO also characterizes XFO_{qr}.

6 The hierarchy $(\mathrm{FO}_q)_{q\in\mathbb{N}}$ on arithmetical structures

Let $\tau_0 := \{<, +, \times\}$ and let τ be a vocabulary with $\tau_0 \subseteq \tau$. For $q \in \mathbb{N}$ by $\mathrm{FO}_q[\tau] \subsetneq \mathrm{FO}_{q+1}[\tau]$ *on arithmetical structures* we mean that there is an $\mathrm{FO}_{q+1}[\tau]$-sentence which is not equivalent to any $\mathrm{FO}_q[\tau]$-sentence on all finite τ-structures with built-in addition and multiplication. We say that *the hierarchy* $(\mathrm{FO}_q)_{q\in\mathbb{N}}$ *is strict on arithmetical structures* if there is a vocabulary $\tau \supseteq \tau_0$ such that $\mathrm{FO}_q[\tau] \subsetneq \mathrm{FO}_{q+1}[\tau]$ on arithmetical structures for every $q \in \mathbb{N}$.

▶ **Theorem 18.** *The hierarchy* $\big(\mathrm{FO}_q\big)_{q\in\mathbb{N}}$ *is strict on arithmetical structures.*

Some preparations are in order. First, we recall how structures are represented by strings. Let τ be a relational vocabulary and $n \in \mathbb{N}$. We encode a τ-structure $\mathcal{A}$ with $A = [n]$ by a binary string $\mathrm{enc}(\mathcal{A})$ of length $\ell_{\tau,n} := \sum_{R\in\tau} n^{\mathrm{arity}(R)}$. For instance, assume $\tau = \{E, P\}$ with binary E and unary P, then $\mathrm{enc}(A) = i_0 i_1 \cdots i_{n^2-1}\, j_0 j_1 \cdots j_{n-1}$ where for every $a, b \in [n]$, $\big(i_{a+b\cdot n} = 1 \iff (a,b) \in E^{\mathcal{A}}\big)$ and $(j_a = 1 \iff a \in P^{\mathcal{A}})$.

Let $\mathbf{K}$ be a class of τ-structures. A family of circuits $(\mathsf{C}_n)_{n\in\mathbb{N}}$ *decides* $\mathbf{K}$ if

1. every C_n has $\ell_{\tau,n}$ inputs,
2. for $n \in \mathbb{N}$ and every τ-structure $\mathcal{A}$ with $A = [n]$, $(\mathcal{A} \in \mathbf{K} \iff \mathsf{C}_n(\mathrm{enc}(\mathcal{A})) = 1)$.

Recall that for $n \in \mathbb{N}$ the classes Σ_n and Π_n of formulas are defined as follows: Σ_0 and Π_0 are the class of quantifier free formulas. The class Σ_{n+1} (the class Π_{n+1}) is the class of formulas of the form $\exists x_1 \ldots \exists x_k \varphi$ with $\varphi \in \Pi_n$ and arbitrary k (of the form $\forall x_1 \ldots \forall x_k \varphi$ with $\varphi \in \Sigma_n$ and arbitrary k). The proof of the following fact is simple.

▶ **Lemma 19.** *Every* FO*-formula of quantifier rank* q *is logically equivalent to a* Σ_{q+1}*-formula and to a* Π_{q+1}*-formula.*

▶ **Lemma 20.** *Let* $q \in \mathbb{N}$*. Then for every sentence* $\varphi \in \mathrm{FO}_q$ *there is a family of circuits* $(\mathsf{C}_n)_{n\in\mathbb{N}}$ *of depth* $\le q+2$ *and size* $n^{O(1)}$ *which decides* $\mathrm{Mod}(\varphi) = \{\mathcal{A} \mid \mathcal{A} \models \varphi\}$*. Moreover, the output of* C_n *is an* OR *gate, and the bottom layer of gates in* C_n *has fan-in bounded by a constant which only depends on* φ*.*

Proof. For notational simplicity we assume $q = 3$. By Lemma 19 the sentence φ is equivalent to a Σ_4-sentence

$$\psi = \exists x_{1,1} \cdots \exists x_{1,i_1} \forall x_{2,1} \cdots \forall x_{2,i_2} \exists x_{3,1} \cdots \exists x_{3,i_3} \forall x_{4,1} \cdots \forall x_{4,i_4} \bigwedge_{p\in I_\wedge} \bigvee_{q\in I_\vee} \chi_{pq},$$

where $I_\wedge$ and $I_\vee$ are index sets and every χ_{pq} is a literal.

For $n \in \mathbb{N}$ we construct the desired circuit $\mathsf{C} = \mathsf{C}_n$ using the standard translation from FO-sentences to AC^0-circuits. That is, every existential (universal) quantifier corresponds to a $\bigvee$ ($\bigwedge$) gate with fan-in n; the conjunction is translated to a $\bigwedge$ gate with fan-in $|I_\wedge|$ and the disjunctions to $\bigvee$ gates with fan-in $|I_\vee|$. Next we merge consecutive layers of gates that are all $\bigwedge$, or that are all $\bigvee$. The resulting circuit C_n is of depth $q+2$. It has an OR as output gate and bottom fan-in bounded by $|I_\vee|$. ◀

Key to our proof of Theorem 18 are the following boolean functions called *Sipser functions*.

▶ **Definition 21** ([16, 5])**.** Let $d \ge 1$ and $m_1, \ldots, m_d \in \mathbb{N}$. For every $i_1 \in [m_1]$, $i_2 \in [m_2]$, $\ldots$, $i_d \in [m_d]$ we introduce a boolean variable $X_{i_1,\ldots,i_d}$. Define

$$f_d^{m_1,\ldots,m_d} := \bigwedge_{i_1\in[m_1]} \bigvee_{i_2\in[m_2]} \cdots \bigodot_{i_d\in[m_d]} X_{i_1,\ldots,i_d}, \tag{7}$$

where $\bigodot$ is $\bigvee$ if d is even, and $\bigwedge$ otherwise. For every $d \geq 2$ and $m \geq 1$ we set

$$\mathrm{Sipser}_d^m := f_d^{m_1,\dots,m_d}$$

with $m_1 = \left\lceil \sqrt{m/\log\ m} \right\rceil$, $m_2 = \cdots m_{d-1} = m$, and $m_d = \left\lceil \sqrt{d/2 \cdot m \cdot \log\ m} \right\rceil$.

Observe that the size of Sipser_d^m is bounded by $m^{O(d)}$.

The following lower bound for Sipser_d^m is proved in [11]. We use the version presented as Theorem 4.2 in [15].

▶ **Theorem 22.** *Let $d \geq 2$. Then there exists a constant $\beta_d > 0$ so that if a depth $d+1$, bottom fan-in k circuit with an* OR *gate as the output and at most S gates in levels* 1 *through d computes Sipser_d^m, then either $S \geq 2^{m^{\beta_d}}$ or $k \geq m^{\beta_d}$.*

Proof of Theorem 18. $\mathrm{FO}_0 \subsetneq \mathrm{FO}_1$ is trivial by considering the sentence $\exists x\, Ux$ where U is a unary relation symbol. We still need to show that for an appropriate vocabulary $\tau \supseteq \tau_0$ it holds $\mathrm{FO}_q[\tau] \subsetneq \mathrm{FO}_{q+1}[\tau]$ on arithmetical structures for every $q \geq 1$.

Let $d, m \in \mathbb{N}$. We identify the function Sipser_d^m with the circuit in (7) which computes it. Let E be a binary relation symbol and U a unary relation symbol. Then we view the underlying (directed) graph of Sipser_d^m as a $\{E, U\}$-structure $\mathcal{A}_{d,m}$ with

$$A_{d,m} := \{v_g \mid g \text{ a gate in } \mathrm{Sipser}_d^m\}, \qquad E^{\mathcal{A}_{d,m}} := \{(v_{g'}, v_g) \mid g' \text{ is an input to } g\},$$
$$U^{\mathcal{A}_{d,m}} := \{v_g \mid g \text{ is an input to the output gate}\}.$$

Let P be a unary relation symbol. Every assignment B of (truth values to the input nodes of) Sipser_d^m can be identified with $P^{\mathcal{A}_{d,m}} := \{g \mid g \text{ an input gate assigned to } \textsc{true} \text{ by } B\}$. For $\tau' := \{E, U, P)$ we define an $\mathrm{FO}[\tau']$-sentence φ_d such that for all m,

$$\mathrm{Sipser}_d^m(P^{\mathcal{A}_{d,m}}) = \textsc{true} \quad \Longleftrightarrow \quad (\mathcal{A}_{d,m}, P^{\mathcal{A}_{d,m}}) \models \varphi_d. \tag{8}$$

Fix $q \geq 1$. Assume q is even and set $d := q+1$ (the case of odd q is treated similarly). We define inductively $\mathrm{FO}[\tau']$-formulas $\psi_\ell(x)$ by

$$\psi_0(x) := Px, \qquad \text{and} \qquad \psi_{\ell+1}(x) := \begin{cases} \forall y\big(Eyx \to \psi_\ell(y)\big) & \text{if } \ell \text{ is even,} \\ \exists y\big(Eyx \wedge \psi_\ell(y)\big) & \text{if } \ell \text{ is odd.} \end{cases}$$

We set (recall the definition of $U^{\mathcal{A}_{d,m}}$)

$$\varphi_{q+1} := \forall x(Ux \to \psi_q(x)).$$

It is straightforward to verify that $\mathrm{qr}(\varphi_{q+1}) = q+1$ and that φ_{q+1} satisfies (8) (for $d = q+1$).

Let $\tau := \tau' \cup \{<, +, \times\} = \{E, U, P, <, +, \times\}$. We define

$$\textsc{Sipser}_{q+1} := \left\{\mathcal{A} \in \mathrm{ARITHM}[\tau] \;\middle|\; \mathcal{A} \models \varphi_{q+1}\right\}.$$

By definition the class $\textsc{Sipser}_{q+1}$ is axiomatizable in $\mathrm{FO}_{q+1}[\tau]$. We show that $\textsc{Sipser}_{q+1}$ is not axiomatizable in $\mathrm{FO}_q[\tau]$. For a contradiction, assume that $\textsc{Sipser}_{q+1} = \mathrm{Mod}(\varphi)$ for some $\varphi \in \mathrm{FO}_q[\tau]$. Then by Lemma 20 there exists a family of circuits $\big(\mathsf{C}_n\big)_{n\in\mathbb{N}}$ such that the following conditions are satisfied.

(C1) Every C_n has $\ell_{\tau,n}$ inputs, depth $q+2$, and size $\ell_{\tau,n}^{O(1)}$.

(C2) The output of C_n is an OR gate, and its bottom fan-in is bounded by a constant.

(C3) For every $n \in \mathbb{N}$ and every τ-structure $\mathcal{A}$ with $A = [n]$

$$\mathcal{A} \in \textsc{Sipser}_{q+1} \quad \Longleftrightarrow \quad \mathsf{C}_n(\mathrm{enc}(\mathcal{A})) = 1.$$

Let $m \in \mathbb{N}$ and let n be the number of variables in Sipser^m_{q+1}, i.e.,

$$n = \left\lceil \sqrt{m/\log m} \right\rceil \cdot m^{q-1} \cdot \left\lceil \sqrt{(q+1)/2 \cdot m \cdot \log m} \right\rceil .$$

Consider the structure $\mathcal{A}_{q+1,m}$ associated with Sipser^m_{q+1} and expand it with $<, +, \times$. Thus for any assignment of the n inputs, identified with the unary relation $P^{\mathcal{A}_{q+1,m}}$, we have

$$\begin{aligned} \mathrm{Sipser}^m_{q+1}(P^{\mathcal{A}_{q+1,m}}) = 1 \quad &\iff \quad (\mathcal{A}_{q+1,m}, <, +, \times, P^{\mathcal{A}_{q+1,m}}) \models \varphi \\ &\iff \quad \mathsf{C}_n\Big(\mathrm{enc}\,(\mathcal{A}_{q+1,m}, <, +, \times, P^{\mathcal{A}_{q+1,m}})\Big) = 1. \end{aligned}$$

Here is the crucial observation. In the string $\mathrm{enc}(\mathcal{A}_{q+1,m}, <, +, \times, P^{\mathcal{A}_{q+1,m}})$ only the last n bits depend on the assignment, that is, on $P^{\mathcal{A}_{q+1,m}}$. These are precisely the n input bits for the Sipser^m_{q+1} function. Thus we can simplify the circuit C_n by fixing the values of the first $\ell_{\tau,n} - n$ inputs according to $(\mathcal{A}_{q+1,m}, <, +, \times)$. Let C^*_n be the resulting circuit. We have

$$\mathrm{Sipser}^m_{q+1}(P^{\mathcal{A}_{q+1,m}}) = 1 \quad \iff \quad \mathsf{C}^*_n(P^{\mathcal{A}_{q+1,m}}) = 1.$$

By (C1), C^*_n has depth $q+2$ and size $n^{O(1)}$ (as $\ell_{\tau,n} = n^{O(1)}$). By (C2) its output is an OR gate, and its bottom fan-in is bounded by a constant. As $m \in \mathbb{N}$ is arbitrary, this clearly contradicts Theorem 22. ◀

Proof of Theorem 2. Let $q \in \mathbb{N}$. By Theorem 18 we know that there is a vocabulary τ and an $\mathrm{FO}_{q+1}[\tau]$-sentence φ which is not equivalent to any $\mathrm{FO}_q[\tau]$-sentences on arithmetical structures. We claim that

$$Q := \big\{(\mathcal{A}, 0) \;\big|\; \mathcal{A} \in \mathrm{ARITHM}[\tau] \text{ and } \mathcal{A} \models \varphi\big\}$$

is not slicewise definable in FO_q. As Q is slicewise definable in FO_{q+1}, this would give us the desired separation.

Assume otherwise, then, by Definition 6, there is a constant $m_0 \in \mathbb{N}$ and a sentence ψ in $\mathrm{FO}_q[\tau \cup C(m_0)]$ such that for every $\mathcal{A} \in \mathrm{ARITHM}[\tau]$

$$\mathcal{A} \models \varphi \quad \iff \quad \mathcal{A}_{C(m_0)} \models \psi.$$

This does not give us a contradiction immediately, since ψ might contain constants in $C(m_0)$. But it is easy to see that Lemma 19 and Lemma 20 both survive in the presence of constants. Thus almost the same proof of Theorem 18 shows that $\psi \in \mathrm{FO}_q[\tau \cup C(m_0)]$ cannot exist. ◀

7 Conclusions

We have shown that a few parameterized problems are slicewise definable in first-order logic with bounded quantifier rank. In particular, the k-vertex-cover problem, i.e., the kth slice of p-Vertex-Cover, is definable in FO_{17} for every $k \in \mathbb{N}$. One natural follow-up question is whether this is optimal. Or can we show at least that p-Vertex-Cover $\notin \mathrm{XFO}_2$? Such a question is reminiscent of the recent quest for optimal algorithms for natural polynomial time solvable problems (see e.g., [2]). In our result p-d-Hitting-Set $\in \mathrm{XFO}_q$ we have $q = O(d^2)$, and we conjecture that there is no universal constant q which works for every p-d-Hitting-Set. But so far, we do not know how to prove such a result.

It turns out that the class $\mathrm{XFO}_{\mathrm{qr}}$ coincides with the parameterized circuit complexity class para-AC^0 which has been intensively studied in [3, 6]. Similar to [3], it seems that all the non-trivial examples in $\mathrm{XFO}_{\mathrm{qr}}$ require the color coding technique. It would be interesting to see whether other tools from parameterized complexity can be used to show membership in $\mathrm{XFO}_{\mathrm{qr}}$.

We have also established the strictness of $\big(\mathrm{XFO}_q\big)_{q\in\mathbb{N}}$ by proving that $\mathrm{FO}_q \subsetneq \mathrm{FO}_{q+1}$ on arithmetical structures for every $q \in \mathbb{N}$. Our proof is built on a strict AC^0-hierarchy on Sipser functions. We conjecture that the sentence

$$\exists x_1 \cdots \exists x_{q+1} \bigwedge_{1\le i<j\le q+1} E_{x_i x_j},$$

which characterizes the existence of a $(q+1)$-clique, witnesses $\mathrm{FO}_q \subsetneq \mathrm{FO}_{q+1}$ on graphs with built-in addition and multiplication. Rossman [14] has shown that $(q+1)$-clique cannot be expressed in arithmetical structures with $\lfloor (q+1)/4 \rfloor$ variables and hence not in $\mathrm{FO}_{\lfloor (q+1)/4 \rfloor}$. This already shows that the hierarchy $(\mathrm{FO}_q)_{q\in\mathbb{N}}$ does not collapse.

References

1. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.
2. A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.
3. M. Bannach, C. Stockhusen, and T. Tantau. Fast parallel fixed-parameter algorithms via color coding. In *Proceedings of the 10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 224–235, 2015.
4. D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.
5. R. B. Boppana and M. Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 757–804. MIT Press, 1990.
6. Y. Chen and J. Flum. Some lower bounds in parameterized AC^0. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 – Kraków, Poland*, pages 27:1–27:14, 2016.
7. M. Elberfeld, C. Stockhusen, and T. Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015.
8. J. Flum and M. Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM Journal on Computing*, 31(1):113–145, 2001.
9. J. Flum and M. Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.
10. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
11. J. Håstad. Almost optimal lower bounds for small depth circuits. In *Randomness and Computation*, pages 6–20. JAI Press, 1989.
12. N. Immerman. *Descriptive Complexity*. Graduate Texts in Computer Science. Springer, 1999.
13. R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 1(1):89–102, 2003.
14. B. Rossman. On the constant-depth complexity of k-clique. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008, Victoria, British Columbia, Canada*, pages 721–730, 2008.
15. N. Segerlind, S. R. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004.
16. M. Sipser. Borel sets and circuit complexity. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 61–69, 1983.

Integral Categories and Calculus Categories*

Robin Cockett[1] **and Jean-Simon Lemay**[2]

1 **Dept. of Computer Science, University of Calgary, Calgary, AB, Canada**
robin@ucalgary.ca

2 **Dept. of Mathematics and Statistics, University of Calgary, Calgary, AB, Canada**
jeansimon.lemay@ucalgary.ca

Abstract

Differential categories are now an established abstract setting for differentiation. The paper presents the parallel development for integration by axiomatizing an integral transformation, $\mathsf{s}_A : !A \to !A \otimes A$, in a symmetric monoidal category with a coalgebra modality. When integration is combined with differentiation, the two fundamental theorems of calculus are expected to hold (in a suitable sense): a differential category with integration which satisfies these two theorem is called a *calculus category*.

Modifying an approach to antiderivatives by T. Ehrhard, it is shown how examples of calculus categories arise as differential categories with antiderivatives in this new sense. Having antiderivatives amounts to demanding that a certain natural transformation, $\mathsf{K} : !A \to !A$, is invertible. We observe that a differential category having antiderivatives, in this sense, is always a calculus category and we provide examples of such categories.

1998 ACM Subject Classification F.3.3 Studies of Program Constructs, F.4.1 Mathematical Logic

Keywords and phrases Differential Categories, Integral Categories, Calculus Categories

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.20

1 Introduction

The two fundamental theorems of calculus relate the two most important operations of calculus: differentiation and integration. The first theorem states that the derivative of the integral of a real function f is the original function: $\frac{\mathsf{d}(\int_a^t f(x)\ \mathsf{d}x)}{\mathsf{d}t}(x) = f(x)$. While the second states that the integral of the derivative of a real function f on a closed interval $[a, b]$ is equal to the difference of f evaluated at the end points: $\int_a^b \frac{\mathsf{d}f(t)}{\mathsf{d}t}(x)\ \mathsf{d}t = f(b) - f(a)$. They are called "fundamental" theorems because they are absolutely fundamental to the development of classical calculus.

Since the turn of the 21st century, there has been significant progress in the abstract understanding of differentiation with the study of differential categories. The abstract formulation of integration, on the other hand, has not received the same level of attention. Nonetheless, one might expect that, when suitably adjoined to the formulation of differentiation, a commensurate abstract form for integration should encompass these fundamental theorems. The purpose of this extended abstract is to explore the extent to which this expectation is realized.

* This work was partially supported by NSERC, Canada.

In the early 2000's, T. Ehrhard and L. Regnier introduced the differential λ-calculus [12] and differential proof nets [13], which formalized differentiation in linear logic. A few years later, R. Blute, R. Cockett and R. Seely introduced differential categories [7], which were the appropriate categorical structure for modelling Ehrhard and Regnier's differential linear logic. Differential categories now have a rich literature of their own [4, 3, 2, 6, 14, 10, 9] and there are many examples which have been extensively studied [7, 11, 5]. However, as mentioned above, little attention has been given to abstracting integration.

In 2014, T. Ehrhard observed that in certain $*$-autonomous categories which had the appropriate structure to be a differential category, it was possible with one additional assumption to produce *antiderivatives* [11]. The additional assumption was that a certain natural transformation – which he called J – constructed from the deriving transformation was a natural isomorphism. With this assumption, Ehrhard constructed an integral transformation with an inverse behaviour to the deriving transformation, in the sense that he gave necessary and sufficient conditions for a *map* to satisfy the first fundamental theorem of calculus – by proving Poincaré's Lemma. Furthermore, when the deriving transformation satisfied an extra – non-equational – condition, which he called the "Taylor Property", he then showed that every differentiable function satisfied the second fundamental theorem of calculus.

While much of the inspiration for our approach to integration derives from these observations, Ehrhard made no attempt to axiomatize integration separately from differentiation. Here we introduce (tensor) integral category as a notion which stands on its own (i.e. in the absence of differentiation). The inspiration for this independent axiomatization of integral categories comes from the much older notion of a Rota-Baxter algebra [1, 20, 15], the classical algebraic abstraction of integration. Briefly, for a commutative ring R and $\lambda \in R$, a Rota-Baxter algebra of weight λ is an R-algebra A with an R-linear morphism $P : A \to A$ which satisfies the Rota-Baxter rule: $P(a)P(b) = P(aP(b)) + P(P(a)b) + \lambda P(ab)$ for all $a, b \in A$. The map P is called a Rota-Baxter operator of weight λ. A particular example of a Rota-Baxter algebra of weight zero is the $\mathbb{R}$-algebra of real continuous functions $\mathsf{Cont}(\mathbb{R})$, where the Rota-Baxter operator $P : \mathsf{Cont}(\mathbb{R}) \to \mathsf{Cont}(\mathbb{R})$ is defined as the integral of the function centred at zero: $P(f)(x) = \int_0^x f(t)\ \mathsf{d}t$. The Rota-Baxter rule for this example is the expression of the integration by parts rule without the use of derivatives: $\int_0^x f(t)\ \mathsf{d}t \cdot \int_0^x g(t)\ \mathsf{d}t = \int_0^x f(t) \cdot (\int_0^t g(u)\ \mathsf{d}u)\ \mathsf{d}t + \int_0^x (\int_0^t f(u)\ \mathsf{d}u) \cdot g(t)\ \mathsf{d}t$ (see [15] for more details). This motivates the Rota-Baxter rule as an axiom of integration.

When differentiation and integration are combined into what we call here a *calculus category*, we demand that the two fundamental theorems hold. The second fundamental theorem is assumed to hold verbatim. However, the first fundamental theorem, as above, has to be interpreted as being on *maps* – rather than objects – and, under this interpretation, becomes the Poincaré property, a conditional property which provides necessary and sufficient conditions for a map to be the differential of its integral. The name of the condition comes from the Poincaré Lemma from cohomology [22] and differential topology [8], which states an analogous result of giving criteria for a map to be an antiderivative.

To obtain the notion of integration as an *antiderivative*, we insist that a slightly different natural transformation, which we call K, should be invertible. We show this is equivalent to requiring both that Ehrhard's transformation J is invertible and that the "Taylor Property" – which Ehrhard had suggested was desirable – holds. This improvement is easily underestimated: the "Taylor Property" is a conditional requirement, replacing a conditional requirement by a purely equational requirement is *always*, mathematically, a significant step. Demanding that K is invertible not only produces an integral transformation, but also secures the first *and* second fundamental theorem of calculus. Inverting only Ehrhard's transformation, J,

does not by itself even produce an integral transformation; the "Taylor Property" is required, in addition to the invertibility of J, to secure an integral transformation. The fact that, when K is invertible, J is invertible is useful particularly in the proof of the Poincaré's lemma. Thus, it is important to observe that, the antiderivative produced by the inverse of K is precisely the same as the antiderivative produced by the inverse of J – *when* K *is already invertible*.

Finally, the notion of a differential category with anti-derivatives, given by requiring K to be invertible, provides a plentiful supply of calculus categories as we explain.

Before beginning, we should mention the conventions that we use in the paper. First off, we will use diagrammatic order for composition. Explicitly, this means that the composite map fg is the map which first does f then g. Secondly, to simplify working in symmetric monoidal categories, we will allow ourselves to work in strict symmetric monoidal categories and so will generally suppress the associator and unitor isomorphisms. For a symmetric monoidal category we will use $\otimes$ for the tensor product, I for the monoidal unit, and $\sigma : A \otimes B \to B \otimes A$ for the symmetry isomorphism.

Full detailed proofs of all the results in this extended abstract can be found in the second author's masters thesis [18].

2 Coalgebra Modalities

Tensor integral and differential categories are structures over additive symmetric monoidal categories with a coalgebra modality. We begin by recalling the components of this structure starting with the notion of an additive category. Here we mean "additive" in the sense of being commutative monoid enriched. Thus, we do not assume negatives nor do we assume biproducts (this differs from the usage in [19] for example). This allows many important examples such as the category of sets and relation or the category of modules of a commutative rig[1].

▶ **Definition 1.** An **additive category** is a commutative monoid enriched category, that is a category in which each hom-set is a commutative monoid – with addition operation $+$ and zero 0 – and in which composition preserves addition that is:
[Add.1] $k(f+g) = kf+kg$ and $0f = 0$;
[Add.2] $(f+g)h = fh+gh$ and $f0 = 0$.

An **additive symmetric monoidal category** is an additive category with a tensor product which is compatible with the additive structure in the sense that:
[Add$_\otimes$.1] $(f+g) \otimes h = f \otimes h+g \otimes h$ and $0 \otimes h = 0$;
[Add$_\otimes$.2] $k \otimes (f+g) = k \otimes f+k \otimes g$ and $h \otimes 0 = 0$.

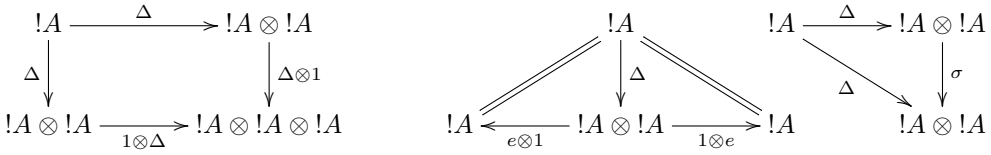
In any additive category their is a notion of "scalar multiplication" of maps by the natural numbers $\mathbb{N}$. The scalar multiplication of a map $f : A \to B$ by $n \in \mathbb{N}$, is the map $n \cdot f : A \to B$ defined by summing n copies of f together. If $n = 0$, then $0 \cdot f$ is simply the zero map from A to B. Furthermore, for additive symmetric monoidal categories, one then has that $(n \cdot f) \otimes g = n \cdot (f \otimes g) = f \otimes (n \cdot g)$.

▶ **Definition 2.** A **coalgebra modality** [7, 3] on a symmetric monoidal category is a quintuple $(!, \delta, \varepsilon, \Delta, e)$ consisting of a comonad $(!, \delta, \varepsilon)$, a natural transformation Δ with

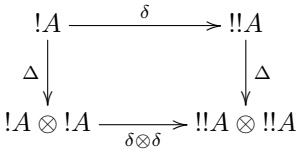
[1] Rigs are also known as a semirings: they are ri*n*gs without *n*egatives.

components $\Delta_A : !A \to !A \otimes !A$, and a natural transformation e with components $e_A : !A \to I$ such that for each object A:

(i) $(!A, \Delta_A, e_A)$ is a cocommutative comonoid, that is, the following diagrams commute:



(ii) δ_A preserves the comultiplication, that is, the following diagram commutes:



When combined with the additive structure, this ensures that $!A$ is a coalgebra in the classical algebraic sense. Furthermore, one can prove that $\delta e = e$ so δ is actually a comonoid homomorphism.

The coKleisli maps for the comonad are important: these maps are of the form $f : !A \to B$: amongst these are the **linear** maps $\varepsilon g : !A \to B$ where $g : A \to B$.

Note that we do not assume that the coalgebra modality, !, is a monoidal functor: to do so would put us in the realm of Seely categories [3, 14] which is more than we require for this basic theory.

3 Integral Categories

Integral categories are the integral analogue of differential categories, thus, the main ingredient of an integral category is an integral transformation, $\mathsf{s}_A :!A \to !A \otimes A$, a natural transformation opposite in orientation to a deriving transformation which must satisfy just three equations:

▶ **Definition 3.** An additive symmetric monoidal category with a coalgebra modality is an **integral category** if there is a natural transformation $\mathsf{s}_A : !A \to !A \otimes A$, called the **integral transformation**, satisfying the following equations:

[s.1] Constants Rule: $\mathsf{s}(e \otimes 1) = \varepsilon$

[s.2] Rota-Baxter Rule: $\Delta(\mathsf{s} \otimes \mathsf{s}) = \mathsf{s}(\Delta \otimes 1)(\mathsf{s} \otimes 1 \otimes 1) + \mathsf{s}(\Delta \otimes 1)(1 \otimes \sigma)(1 \otimes 1 \otimes \mathsf{s})$

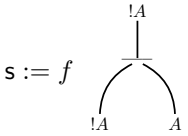
[s.3] Interchange Rule: $\mathsf{s}(\mathsf{s} \otimes 1) = \mathsf{s}(\mathsf{s} \otimes 1)(1 \otimes \sigma)$

The integral of a map $f : !A \otimes A \to B$ is defined as the composition $\mathsf{S}[f] := \mathsf{s}_A f : !A \to B$. This should be thought of as the classical integral of f evaluated from 0 to x as a function of x: $\mathsf{S}[f](x) := \int_0^x f(t)\ \mathsf{d}t$. To interpret this as $\mathsf{S}[f]$ one must regard f as being a function of two variables t and $\mathsf{d}t$, which is linear in $\mathsf{d}t$. Classically, f is regarded as a function of one (one dimensional) variable, t, and to obtain the interpretation as a function of two arguments one simply multiplies by the variable $\mathsf{d}t$. This allows a simple interpretation of the integral notation for one dimensional functions: it leaves open the interpretation for multidimensional functions – an issue to which we shall return.

The additive structure of the category ensures the integral of a sum of maps is equal to the sum of the integral of each map, that is, $\mathsf{S}[f + g] = \mathsf{S}[f] + \mathsf{S}[g]$ and $\mathsf{S}[0] = 0$. The first axiom **[s.1]** states that the integral of a constant map is a linear map (in the sense discussed above). The second axiom **[s.2]** is the Rota-Baxter rule [15], which is an expression

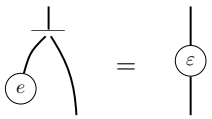
of integration by parts using only integrals. In classical calculus notation, the Rota-Baxter rule is expressed as: $\int_0^x f(t)\ \mathsf{d}t \cdot \int_0^x g(t)\ \mathsf{d}t = \int_0^x f(t) \cdot (\int_0^t g(u)\ \mathsf{d}u)\ \mathsf{d}t + \int_0^x (\int_0^t f(u)\ \mathsf{d}u) \cdot g(t)\ \mathsf{d}t$. The third axiom **[s.3]** ensures the independence of the order of integration – the interchange law – that is integrating with respect to u then t is the same as integrating with respect to t then u. It may be tempting to think this is related to Fubini's theorem. In fact, it is not closely related at all: we discuss this at the end of this section. **[s.3]** can be expressed in classical notation as: $\int_0^x (\int_0^t f(u)\ \mathsf{d}u\ \mathsf{d}t) = \int_0^x (\int_0^u f(t)\ \mathsf{d}t\ \mathsf{d}u)$.

Just like differential categories, integral categories have a graphical calculus (see [21] for an introduction to the graphical calculus in monoidal categories and its variations). We represent the integral transformation in string diagrams as follows (which should be read from top to bottom):

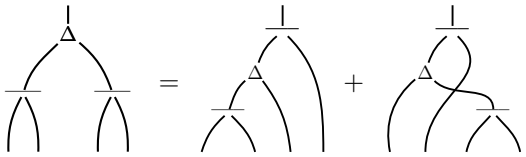


The integral axioms **[s.1]**, **[s.2]** and **[s.3]** are then represented in the graphical calculus as follows (we omit writing the objects at the end of the wires).

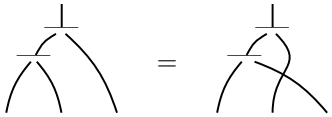
[s.1] Constants Rule:



[s.2] Rota-Baxter Rule:

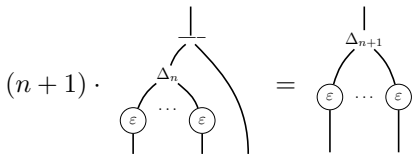


[s.3] Interchange Rule:

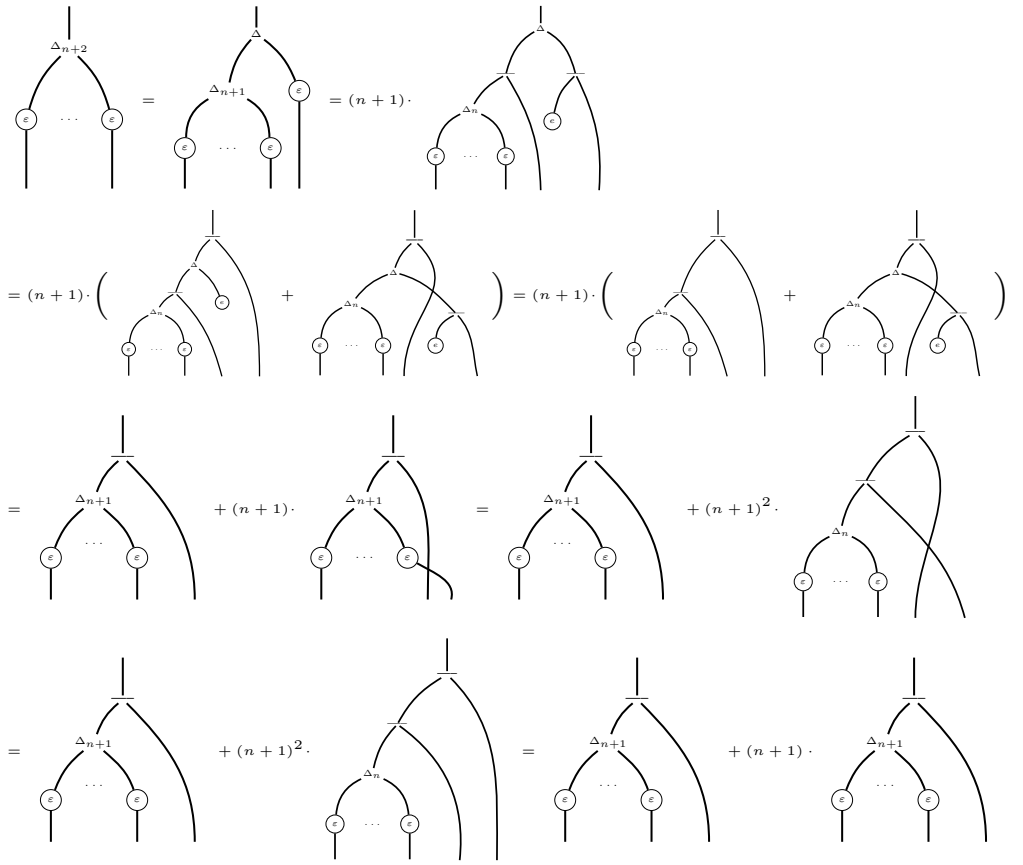


With the graphical calculus, we are now in a position to explore polynomial integration. Perhaps the first formula learnt in first year calculus is $\int_0^x x^n \mathsf{d}x = \frac{1}{n+1} x^{n+1}$. However this formula cannot be expressed in a general additive category – simply because there may not be fractions. That said, we will soon see that in every integral category there is a notion of scalar multiplication by positive rationals, that is, certain hom-sets are $\mathbb{Q}_{\geq 0}$-modules, where $\mathbb{Q}_{\geq 0}$ is the rig of non-negative rationals. The integral of monomials identity can be re-express as the requirement that $(n+1) \int_0^x x^n\ \mathsf{d}x = x^{n+1}$ and this identity does hold in any integral category! To express this identity in an integral category, we will need the n-fold comultiplication $\Delta_n : !A \to !A^{\otimes^n}$ which is defined as $\Delta_n = \Delta(\Delta \otimes 1)(\Delta \otimes 1 \otimes 1)...(\Delta \otimes 1^{\otimes^{n-2}})$. By convention we set $\Delta_0 = e$, $\Delta_1 = 1$ and $\Delta_2 = \Delta$.

▶ **Theorem 4.** *For every $n \in \mathbb{N}$, the integral transformation satisfies the polynomial identity:* **[s. Poly]** $(n+1) \cdot \mathsf{s}(\Delta_n \otimes 1)(\varepsilon^{\otimes^n} \otimes 1) = \Delta_{n+1}(\varepsilon^{\otimes^{n+1}})$



Proof. The beauty of this proof is that it uses every integral transformation axiom. The proof is much smoother using the graphical calculus, which is equivalent to proofs done algebraically as shown in [16]. We will prove the equality for the integral transformation by induction on n. For the base case of $n = 0$, this equality holds directly by the constant rule **[s.1]**. Assume the induction hypothesis **[s. Poly]** holds for n, we now show it for $n + 1$:



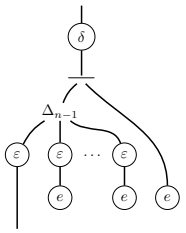
◀

An important consequence of polynomial integration is that certain hom-sets are $\mathbb{Q}_{\geq 0}$-modules. In an additive category, for every object A and for every natural number $n \in \mathbb{N}$, define the map $n_A : A \to A$ by summing n copies of the identities: $n_A = n \cdot 1_A$. We will now prove that in any integral category that for every object A and $n \geq 2$, the map $n_{!A}$ is invertible.

▶ **Theorem 5.** *In an integral category, for every natural number $n \in \mathbb{N}$, $n \geq 2$, and every object $A \in \mathbb{X}$, the map $n_{!A} : !A \to !A$ is an isomorphism.*

▶ Remark. Notice that the case $n = 1$ is also true since the identity map is an isomorphism.

Proof. We will simply define the inverse of $n_{!A}$. For each object A and $n \geq 2$, define $n_{!A}^{-1} : !A \to !A$, as: $n_{!A}^{-1} = \delta_A \mathsf{s}_{!A}(\Delta_{n-1} \otimes 1)(\varepsilon_{!A}^{\otimes^{n-1}} \otimes 1)(1 \otimes e^{\otimes^{n-1}})$, written in the graphical calculus as:

$n_{!A}^{-1} =$ 

◀

This implies, in an integral category, hom-sets with domain $!A$ are $\mathbb{Q}_{\geq 0}$-modules. The scalar multiplication of a map $f : !A \to B$ with a non-negative rational $\frac{p}{q} \in \mathbb{Q}_{\geq 0}$ is the map $\frac{p}{q} \cdot f : !A \to B$ defined as $\frac{p}{q} \cdot f = q_{!A}^{-1}(p \cdot f)$.

Finally, we discuss the interpretation of Fubini's theorem. The theorem requires that the coalgebra modality is monoidal and, thus, that there is a Seely isomorphism [2, 14]: $\chi : !A \otimes !B \to !(A \times B)$. Fubini's theorem concerns the double integration of a function of the form $f : !(A \times B) \otimes A \otimes B \to C$ whose type ensures it is bilinear in the second two occurrences of A and B. Functions of this form can be integrated with respect to either A or B, or both A and B: the latter, the double integral of f, is obtained as follows:

$$!(A \times B) \xrightarrow{\chi^{-1}} !A \otimes !B \xrightarrow{s \otimes s} !A \otimes A \otimes !B \otimes B \xrightarrow{1 \otimes \sigma \otimes 1} !A \otimes !B \otimes A \otimes B \xrightarrow{\chi \otimes 1 \otimes 1} !(A \times B) \otimes A \otimes B \xrightarrow{f} C$$

Fubini's theorem asserts that the order of integration in this double integral does not matter. At this level of generality this order independence is an immediate consequence of the bifunctoriality of $\_ \otimes \_$.

4 Calculus Categories

In this section we wish to put integration together with differentiation and to discuss how they should interact. We start by briefly recalling the definition of a differential category [7] before introducing calculus categories whose structure is induced by the fundamental theorems of calculus.

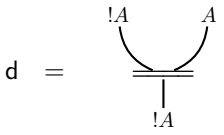
▶ **Definition 6.** An additive symmetric monoidal category with a coalgebra modality is a **differential category** if the coalgebra modality comes equipped with a **deriving transformation** [7], that is, a natural transformation d with components $\mathsf{d}_A : !A \otimes A \to !A$, satisfying the following equations:

[d.1] Constant Rule: $\mathsf{d}e = 0$
[d.2] Leibniz Rule: $\mathsf{d}\Delta = (\Delta \otimes 1)(1 \otimes \sigma)(\mathsf{d} \otimes 1) + (\Delta \otimes 1)(1 \otimes \mathsf{d})$
[d.3] Linear Rule: $\mathsf{d}\varepsilon = (e \otimes 1)\lambda$
[d.4] Chain Rule: $\mathsf{d}\delta = (\Delta \otimes 1)(\delta \otimes 1 \otimes 1)(1 \otimes \mathsf{d})\mathsf{d}$
[d.5] Interchange Rule: $(\mathsf{d} \otimes 1)\mathsf{d} = (1 \otimes \sigma)(\mathsf{d} \otimes 1)\mathsf{d}$

The derivative of a map $f : !A \to B$ is the composition $\mathsf{D}[f] := \mathsf{d}_A f : !A \otimes A \to B$. The first axiom, **[d.1]**, states that the derivative of a constant map is zero. The second axiom **[d.2]** is the Leibniz rule for differentiation – also called the product rule. The third axiom **[d.3]** says that the derivative of a linear map is a constant. The fourth axiom **[d.4]** is the chain rule. The last axiom **[d.5]** is the independence of differentiation or the interchange law, which naively states that differentiating with respect to x then y is the same as differentiation with respect to y then x. It should be noted that **[d.5]** was not a requirement in [7] but

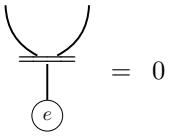
was later added to the definition [3, 3] to ensure that the coKleisli category of a differential category was a Cartesian differential category.

As previously stated, differential categories have a graphical calculus. The deriving transformation is represented as follows:

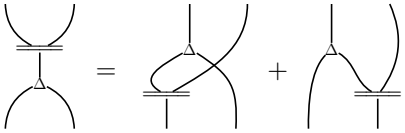


The string diagram representations of **[d.1]** to **[d.5]** are as follows:

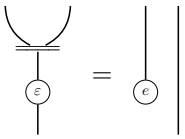
[d.1] Constant Rule:



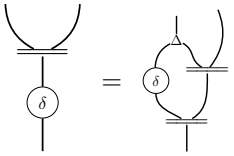
[d.2] Leibniz Rule:



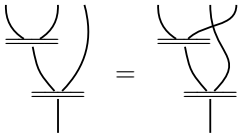
[d.3] Linear Rule:



[d.4] Chain Rule:



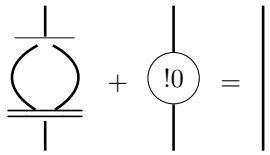
[d.5] Interchange Rule:



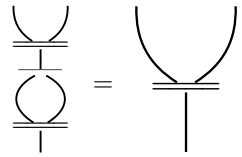
We are now ready to tackle the interaction between integration and differentiation. We start with the second fundamental theorem of calculus and return to discuss the first fundamental theorem of calculus:

▶ **Definition 7.** Let $\mathbb{X}$ be a differential category and an integral category with deriving transformation d and integrating transformation s on the same coalgebra modality $(!, \delta, \varepsilon, \Delta, e)$.

(i) d and s are said to satisfy the **Second Fundamental Theorem of Calculus** if: $\mathsf{sd} + !0 = 1$, written in the graphical calculus as:



(ii) d and s are said to be **compatible** if: $\mathsf{dsd} = \mathsf{d}$, written in the graphical calculus as:



(iii) d is said to be **Taylor** if for every pair of maps $f, g : C \otimes !A \to B$, such that

$$(1 \otimes \mathsf{d})f = (1 \otimes \mathsf{d})g$$

then $f + (1 \otimes !(0))g = g + (1 \otimes !(0))f$.

The first part of the definition expresses the second fundamental theorem of calculus. Compatibility is a weaker version of the second fundamental theorem. The Taylor property (see [11]) is the property that if two maps have the same derivative then they differ by constants.

▶ **Theorem 8.** *For a deriving transformation* d *and an integral transformation* s *on the same coalgebra modality, the following are equivalent:*
(i) d *and* s *satisfy the Second Fundamental Theorem of Calculus;*
(ii) d *and* s *are compatible and* d *is Taylor.*

▶ Remark. This is an extension of Proposition 14 of [11] which proved (ii) ⇒ (i) for Ehrhard's original integral using J^{-1}, however the notion of compatibility was not identified although it was used in the proof.

Proof. (i) ⇒ (ii): Suppose d and s satisfy the Second Fundamental Theorem of Calculus. For Taylor, suppose that $(1 \otimes \mathsf{d})f = (1 \otimes \mathsf{d})g$. Then we have the following equality:

$$f+(1\otimes!0)g = (1\otimes\mathsf{s})(1\otimes\mathsf{d})f+(1\otimes!0)f+(1\otimes!0)g = (1\otimes\mathsf{s})(1\otimes\mathsf{d})g+(1\otimes!0)f+(1\otimes!0)g = g+(1\otimes!0)f$$

For compatibility, by naturality, we have the following equality:

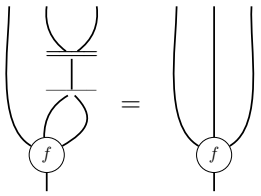
$$\mathsf{d} = \mathsf{dsd} + \mathsf{d}!(0) = \mathsf{dsd} + (!(0) \otimes 0)\mathsf{d} = \mathsf{dsd} + 0 = \mathsf{dsd}\,.$$

(i) ⇒ (ii): Suppose d and s are Compatible and d is Taylor. Notice by Compatibility we have: $\mathsf{dsd} = \mathsf{d}$, and then by Taylor (where $f = \mathsf{sd}$ and $g = 1$) we have the following equality: $\mathsf{sd} + !(0) = 1 + !(0)\mathsf{sd}$. However, using naturality, we have:

$$\mathsf{sd} + !(0) = 1 + !(0)\mathsf{sd} = 1 + \mathsf{s}(!(0) \otimes 0)\mathsf{d} = 1 + 0 = 1\,.$$ ◀

The interpretation of the first fundamental theorem of calculus, unlike the second fundamental theorem, is as a property of a map:

▶ **Definition 9.** A map $f : C \otimes !A \otimes A \to B$ satisfies the **First Fundamental Theorem** (in the last two arguments) if $(1 \otimes (\mathsf{d}_A \mathsf{s}_A))f = f$, written in the graphical calculus as:



Thus, if f satisfies the First Fundamental theorem, it may be viewed as the differential of a map – namely the differential of its integral. Clearly not all maps will satisfy the First Fundamental theorem calculus, a necessary condition is:

▶ **Lemma 10.** *If a map, $f : C \otimes !A \otimes A \to B$, satisfies the First Fundamental Theorem, then:* $(1 \otimes 1 \otimes \sigma)(1 \otimes \mathsf{d} \otimes 1)f = (1 \otimes \mathsf{d} \otimes 1)f$.

Proof. As $(1 \otimes \mathsf{ds})f = f$, the interchange rule for the deriving transformation **[d.5]** gives:

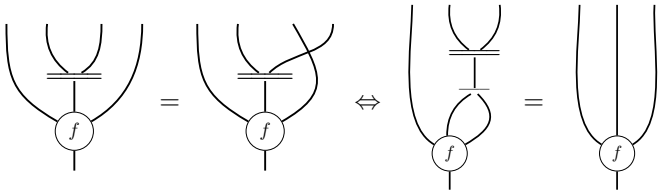
$$(1\otimes 1\otimes\sigma)(1\otimes\mathsf{d}\otimes 1)f = (1\otimes 1\otimes\sigma)(1\otimes\mathsf{d}\otimes 1)(1\otimes(\mathsf{ds}))f = (1\otimes\mathsf{d}\otimes 1)(1\otimes(\mathsf{ds}))f = (1\otimes\mathsf{d}\otimes 1)f$$

◀

We shall use the converse of this lemma as an axiom and call it the Poincaré condition:

▶ **Definition 11.** A differential category with an integral transformation satisfies the **Poincaré condition** if any map $f : C\otimes!A\otimes A \to B$ for which: $(1\otimes 1\otimes\sigma)(1\otimes\mathsf{d}\otimes 1)f = (1\otimes\mathsf{d}\otimes 1)f$, satisfies the First Fundamental Theorem – that is: $(1 \otimes \mathsf{ds})f = f$.

The Poincaré condition and Lemma 10 imply the following equivalence:



The Poincaré condition also implies compatibility of the deriving transformation and integrating transformation.

▶ **Theorem 12.** *The integral and deriving transformation are compatible in any differential category with an integral transformation which satisfies the Poincaré condition.*

Proof. By **[d.5]**, the deriving transformation d satisfies the Poincaré pre-condition that $(1 \otimes \sigma)(\mathsf{d} \otimes 1)\mathsf{d} = (\mathsf{d} \otimes 1)\mathsf{d}$. Therefore, d satisfies the First fundamental theorem of Calculus, which is simply the statement of compatibility: $\mathsf{dsd} = \mathsf{d}$. ◀

▶ **Corollary 13.** *A deriving and integral transformation which satisfy the Poincaré condition such that* d *is Taylor, satisfies the Second Fundamental Theorem of Calculus.*

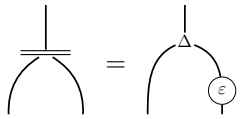
This suggests the following basic definition:

▶ **Definition 14.** A **calculus category** is a differential category and an integral category on the same coalgebra modality such that the deriving transformation and the integral transformation satisfy the Second Fundamental Theorem of Calculus and the Poincaré condition.

5 Antiderivatives

In this last section, we explore how one obtains a calculus category from a differential category with "antiderivatives". A differential category has "antiderivatives" when a certain natural transformation, K – which is present in all differential categories – is a natural isomorphism. This is a strengthening of Ehrhard's original idea in [11], which required a different natural transformation, J, to be a natural isomorphism. Inverting J by itself does not appear to give even an integral category: to obtain an integral category and the second fundamental theorem of calculus Ehrhard also demanded the Taylor property. Inverting K, as we shall see, gets all these properties – and, thus, a calculus category – in one step.

In an additive symmetric monoidal category with a coalgebra modality, the **coderiving transformation** is the natural transformation $\mathsf{d}^\circ_A := \Delta_A(1_{!A} \otimes \varepsilon_A) : !A \to !A \otimes A$ (this called the "annihilation operator" in [14]). We represent the coderiving transformation as an upside down deriving transformation in the graphical calculus:

$\mathsf{d}^\circ :=$  $=$

The coderiving transformation would probably be one's first attempt at constructing an integrating transformation in differential category. The following theorem indicates how close the coderiving transformation is to being an integrating transformation:

▶ **Theorem 15.** *The coderiving transformation* d° *satisfies the following properties:*

[cd.1] $\mathsf{d}^\circ(e \otimes 1) = \varepsilon$

[cd.2] $\mathsf{d}^\circ(\varepsilon \otimes 1) = \Delta(\varepsilon \otimes \varepsilon)$

[cd.3] $\mathsf{d}^\circ(\Delta \otimes 1) = \Delta(1 \otimes \mathsf{d}^\circ)$

[cd.4] $\mathsf{d}^\circ(\Delta \otimes 1)(1 \otimes \sigma) = \Delta(\mathsf{d}^\circ \otimes 1)$

[cd.5] $2 \cdot \Delta(\mathsf{d}^\circ \otimes \mathsf{d}^\circ) = \mathsf{d}^\circ(\Delta \otimes 1)(\mathsf{d}^\circ \otimes 1 \otimes 1) + \mathsf{d}^\circ(\Delta \otimes 1)(1 \otimes \sigma)(1 \otimes 1 \otimes \mathsf{d}^\circ)$

[cd.6] $\mathsf{d}^\circ(\delta \otimes 1) = \delta\mathsf{d}^\circ(1 \otimes \varepsilon)$

[cd.7] $\mathsf{d}^\circ(\mathsf{d}^\circ \otimes 1) = \mathsf{d}^\circ(\mathsf{d}^\circ \otimes 1)(1 \otimes \sigma)$

Notice in particular **[cd.1]**, **[cd.5]** and **[cd.7]**. If we let $\mathsf{s} = \mathsf{d}^\circ$, then **[cd.1]** and **[cd.7]** are precisely the same as **[s.1]** and **[s.3]**. However, the coderiving transformation fails to satisfy **[s.2]**, the Rota-Baxter rule, since **[cd.5]** has an extra factor of 2. Of course, if the differential category is in fact enriched over idempotent commutative monoids, so that $1 + 1 = 1$, the coderiving transformation would be an integral transformation: this happens, for example, in the category of sets and relations.

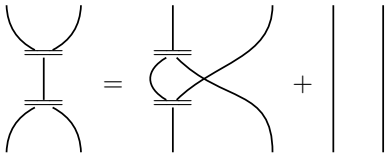
Another important property the coderiving transformation satisfies is its relation with the deriving transformation.

▶ **Theorem 16.** *The deriving and coderiving transformations satisfy the following equality:*

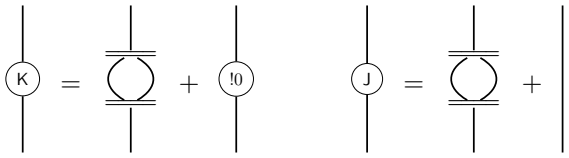
$$\mathsf{d}_A\mathsf{d}^\circ_A = \mathsf{W}_A + 1_{!A \otimes A}$$

where W *is the natural transformation with components* $\mathsf{W}_A = (\mathsf{d}^\circ_A \otimes 1_A)(1_{!A} \otimes \sigma)(\mathsf{d}_A \otimes 1_A)$.

The notation W was introduced by Ehrhard's in [11] where a proof can be found. In the graphical calculus, the above identity is expressed as follows:



In a differential category there are two important natural transformations K and J defined by $\mathsf{K}_A := \mathsf{d}^\circ_A\mathsf{d}_A + !0 : !A \to !A$ and $\mathsf{J}_A := \mathsf{d}^\circ_A\mathsf{d}_A + 1_{!A} : !A \to !A$, written in the graphical calculus as:



K and J satisfy a long list of very similar properties which describe their interaction with the differential structure. We give some of the more important ones in the following theorem:

▶ **Theorem 17.** K *and* J *satisfy the following properties:*

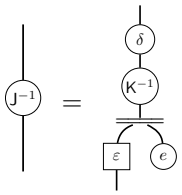
[K.1] $\mathsf{K}!(0) = !(0) = !(0)\mathsf{K}$;
[K.2] $\mathsf{K}e = e$;
[K.3] $\mathsf{K}\varepsilon = \varepsilon$;
[K.4] $\mathsf{K}\Delta = \Delta((\mathsf{dd}^\circ) \otimes 1) + \Delta(1 \otimes (\mathsf{dd}^\circ)) + \Delta(!(0) \otimes !(0))$;
[K.5] $\mathsf{K}\delta = \delta\mathsf{d}^\circ(1 \otimes (\mathsf{dd}^\circ))\mathsf{d} + \delta!(!(0))$;
[K.6] $(\mathsf{K} \otimes 1)\mathsf{W} = \mathsf{W}(\mathsf{K} \otimes 1)$;
[K.7] $(\mathsf{K} \otimes 1)\mathsf{dd}^\circ = \mathsf{dd}^\circ(\mathsf{K} \otimes 1)$.

[J.1] $\mathsf{J}!(0) = !(0) = !(0)\mathsf{J}$;
[J.2] $\mathsf{J}e = e$;
[J.3] $\mathsf{J}\varepsilon = 2 \cdot \varepsilon$;
[J.4] $\mathsf{J}\Delta = \Delta(\mathsf{J} \otimes 1) + \Delta(1 \otimes (\mathsf{dd}^\circ)) = \Delta((\mathsf{dd}^\circ) \otimes 1) + \Delta(1 \otimes \mathsf{J})$;
[J.5] $\mathsf{J}\delta = \delta\mathsf{d}^\circ(1 \otimes (\mathsf{dd}^\circ))\mathsf{d} + \delta$;
[J.6] $(\mathsf{J} \otimes 1)\mathsf{d} = \mathsf{dK}$;
[J.7] $\mathsf{d}^\circ(\mathsf{J} \otimes 1) = \mathsf{Kd}^\circ$;
[J.8] $(\mathsf{J} \otimes 1)\mathsf{W} = \mathsf{W}(\mathsf{J} \otimes 1)$;
[J.9] $(\mathsf{J} \otimes 1)\mathsf{dd}^\circ = \mathsf{dd}^\circ(\mathsf{J} \otimes 1)$.

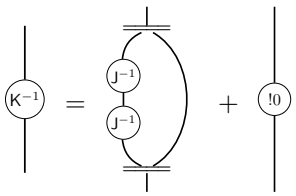
Recall that Ehrhard's original idea was to obtain integration by requiring that J be a natural isomorphism. However, Ehrhard's integral transformation, using only that J is invertible, appears to fail the Rota-Baxter rule **[s.2]**. This is why we have strengthened Ehrhard's approach by requiring instead that K be a natural isomorphism. We observe:

▶ **Theorem 18.** *For a differential category,* K *is a natural isomorphism if and only if* J *is a natural isomorphism and the deriving transformation is Taylor.*

Proof. We give the definitions of K^{-1} and J^{-1}:
(i) $\Rightarrow$ (ii) If K is a natural isomorphism, then: $\mathsf{J}^{-1}_A := \delta_A \mathsf{K}^{-1}_{!A} \mathsf{d}^{\circ}_{!A}(!(\varepsilon_A) \otimes e_A)\rho_{!A}$, written in the graphical calculus as:



(ii) $\Rightarrow$ (i) If J is a natural isomorphism and the deriving transformation is Taylor, then $\mathsf{K}^{-1}_A := \mathsf{d}^{\circ}_A(\mathsf{J}^{-1}_A \otimes 1_A)(\mathsf{J}^{-1}_A \otimes 1_A)\mathsf{d}_A + !0$, where $0 : A \to A$, and written in the graphical calculus as:



◀

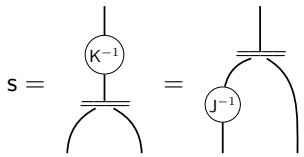
▶ **Definition 19.** A differential category has **antiderivatives** if K is a natural isomorphism.

Equivalently, of course, a differential category has antiderivatives if J is a natural isomorphism and the deriving transformation is Taylor. While our definition of antiderivatives differs only slightly from Ehrhard's, [11], our definition does imply Ehrhard's definition and, at the same time, secures the property of being an integral category for which, as far as we can see, inverting J is insufficient.

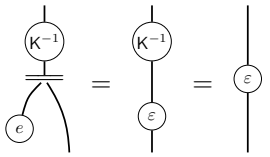
▶ **Theorem 20.** *In a differential category with antiderivatives,* K^{-1} *and* J^{-1} *satisfy the following properties:*
[K^{-1}.1] $\mathsf{K}^{-1}!(0) = !(0) = !(0)\mathsf{K}^{-1}$;
[K^{-1}.2] $\mathsf{K}^{-1}e = e$;
[K^{-1}.3] $\mathsf{K}^{-1}\varepsilon = \varepsilon$;
[K^{-1}.4] $\Delta(\mathsf{K}^{-1} \otimes \mathsf{K}^{-1}) + \Delta(\mathsf{K}^{-1} \otimes !(0)) + \Delta(!(0) \otimes \mathsf{K}^{-1}) = \mathsf{K}^{-1}\Delta(\mathsf{K}^{-1} \otimes 1) + \mathsf{K}^{-1}\Delta(1 \otimes \mathsf{K}^{-1}) + \Delta(!(0) \otimes !(0))$;
[K^{-1}.5] $(\mathsf{K}^{-1} \otimes 1)\mathsf{W} = \mathsf{W}(\mathsf{K}^{-1} \otimes 1)$;
[K^{-1}.6] $(\mathsf{K}^{-1} \otimes 1)\mathsf{dd}^{\circ} = \mathsf{dd}^{\circ}(\mathsf{K}^{-1} \otimes 1)$;

[J^{-1}.1] $\mathsf{J}^{-1}!(0) = !(0) = !(0)\mathsf{J}^{-1}$;
[J^{-1}.2] $\mathsf{J}^{-1}e = e$;
[J^{-1}.3] $2 \cdot \mathsf{J}^{-1}\varepsilon = \varepsilon$;
[J^{-1}.4] $(\mathsf{J}^{-1} \otimes 1)\mathsf{d} = \mathsf{dK}^{-1}$;
[J^{-1}.5] $\mathsf{d}^{\circ}(\mathsf{J}^{-1} \otimes 1) = \mathsf{K}^{-1}\mathsf{d}^{\circ}$;
[J^{-1}.6] $(\mathsf{J}^{-1} \otimes 1)\mathsf{W} = \mathsf{W}(\mathsf{J}^{-1} \otimes 1)$;
[J^{-1}.7] $(\mathsf{J}^{-1} \otimes 1)\mathsf{dd}^{\circ} = \mathsf{dd}^{\circ}(\mathsf{J}^{-1} \otimes 1)$;
In particular, **[J^{-1}.5]** will imply that the integral transformation constructed using either K^{-1} or J^{-1} are equal to one another. Finally, with these properties of K^{-1} and J^{-1}, we obtain the main result of this section, namely that a differential category with antiderivatives is a calculus category:

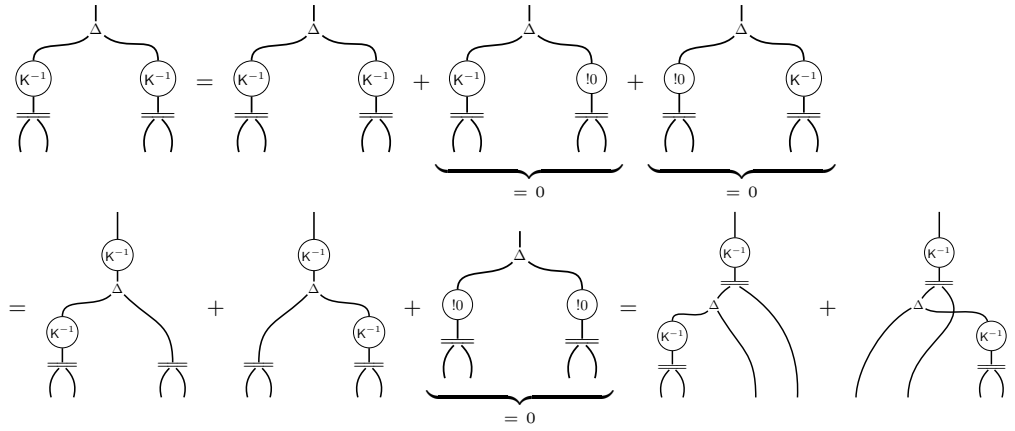
▶ **Theorem 21.** *A differential category with antiderivatives is a calculus category with the integral transformation defined by* $\mathsf{s}_A := \mathsf{K}_A^{-1}\mathsf{d}_A^\circ = \mathsf{d}_A^\circ(\mathsf{J}_A^{-1} \otimes 1_A)$*, expressed in the graphical calculus as:*



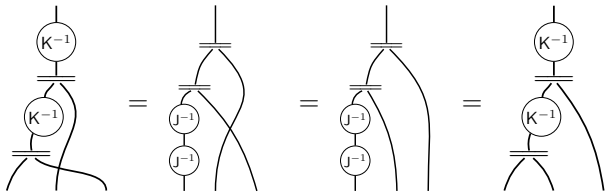
Proof. To prove the integral transformation axioms and the second fundamental theorem we use the K^{-1} form of the integrating transformation. While to prove the Poincaré condition we use Ehrhard's J^{-1}. We will use the graphical calculus to help us. We first show that our integral transformation satisfies [**s.1**] to [**s.3**].
[**s.1**]: Here we use [**cd.1**] and [$\mathbf{K}^{-1}$**.2**]:



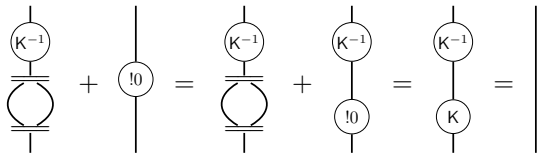
[**s.2**]: Here we use [$\mathbf{K}^{-1}$**.4**], [**cd.3**], [**cd.4**] and naturality of the coderiving transformation:



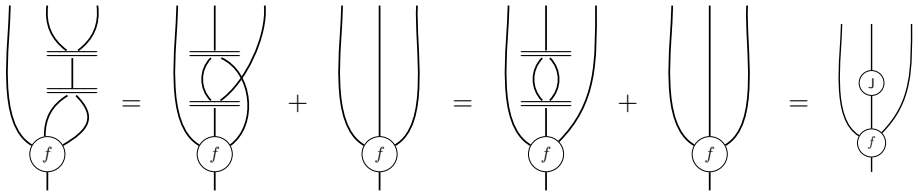
[**s.3**]: Here we use [$\mathbf{J}^{-1}$**.5**] and [**cd.7**]:



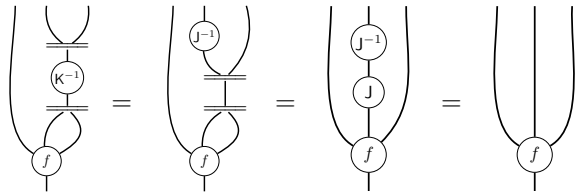
Next we show the second fundamental theorem of calculus. Here we use [$\mathbf{K}^{-1}$**.1**]:



Finally we prove the Poincaré Condition. Let $f : C \otimes !A \otimes A \to B$ satisfy the Poincaré pre-condition, that is, $(1 \otimes 1 \otimes \sigma)(1 \otimes \mathsf{d} \otimes 1)f = (1 \otimes \mathsf{d} \otimes 1)f$. First notice that by Theorem 16 and the Poincaré pre-condition, f satisfies the following identity:



Then using $[\mathbf{J^{-1}.4}]$ and the above identity we get the following equality:



Which completes the proof that antiderivatives give a calculus category. ◀

The converse of Theorem 21 is true if the coalgebra is monoidal (in the sense explained at the end of Section 3 when discussing Fubini's theorem) and the integral transformation is compatible with monoidal strength, that is, a calculus category with a monoidal coalgebra modality and a monoidal integral transformation is a differential category with antiderivatives. More details and a proof of this will be given in a subsequent paper.

We are now are in a position to give two examples of differential categories which have antiderivatives, and therefore, two examples of calculus categories:

▶ **Example 22.** The category of sets and relations, REL, is a differential category [7] with antiderivatives. The symmetric monoidal strucure is given by the Cartesian product of sets while the additive structure is given by the union of sets. The coalgebra modality is given by the finite bag/multiset comonad (see [7] for more details), where for a set X, $!X$ is the set of bags/multisets of X. The deriving transformation $\mathsf{d}_X : !X \times X \to !X$ is the relation which adds an extra element to the bag:

$$\mathsf{d}_X = \{((B, x), B \cup x) |\ x \in X,\ B \in !X\}$$

The additive idempotency of REL makes both K and J the identity and thus trivially isomorphisms. Therefore, the integral transformation is the coderiving transformation $\mathsf{d}^\circ_X : !X \to !X \times X$, which is the relation which removes an elements from the bag:

$$\mathsf{d}^\circ_X = \{(B, (B - \{x\}, x)) |\ x \in X,\ B \in !X\}$$

▶ **Example 23.** The category of vector spaces over a field $\mathbb{K}$ of characteristic of 0, $\mathsf{VEC}_\mathbb{K}$, is a co-differential category [7] with antiderivatives, so that, $\mathsf{VEC}^{\mathsf{op}}_\mathbb{K}$ is a calculus category. While having a field of characteristic zero is not required to obtain differential structure, it is required for antiderivatives. The additive symmetric monoidal structure is given by the standard tensor product and additive enrichment of vector spaces. The algebra modality is given by the free symmetric algebra monad where for a vector space V, $!V$ is the free commutative algebra over V (see [17] for more details). Equivalently, if $X = \{x_1, x_2, ...\}$ is

a basis of V, then $!V$ is isomorphic to the polynomial ring over X: $!V \cong \mathbb{K}[X]$ [17]. Then the deriving transformation $\mathsf{d}_V : !V \to !V \otimes V$ (recall $\mathsf{VEC}_{\mathbb{K}}^{\mathrm{op}}$ is the calculus category) on monomials is given by the sum of partial derivatives of the monomial:

$$\mathsf{d}_V(x_1^{r_1}...x_n^{r_n}) = \sum_{i=1}^{n}(x_1^{r_1}...x_i^{r_i-1}...x^{r_n}) \otimes x_i$$

On monomials, K multiplies the non-constant monomials by their degree and multiplies the constants by one, while J multiplies monomials by their degree plus one. As the rationals are embedded in our field, both are isomorphisms, and the resulting integral transformation $\mathsf{s}_V : !V \otimes V \to !V$ is defined on monomials by:

$$\mathsf{s}_V((x_1^{r_1}...x_n^{r_n}) \otimes x_i) = \frac{1}{1+\sum_{j=1}^{n} r_j} x_1^{r_1}...x_i^{r_i+1}...x^{r_n}$$

At first glance this may seem bizarre. One might expect the integrating transformation to integrate a monomial with respect to the variable x_i and thus only multiply by $\frac{1}{1+r_i}$. However, this classical idea of integration fails the Rota-Baxter rule **[s.2]** for any vector space of dimension greater than one.

6 Conclusion and Future Work

The theory of differential categories was developed in stages: (tensor) differential categories [7], cartesian differential categories [3], differential restriction categories [10], and tangent categories [9]. The development of integral categories, being very closely related, has parallel stages. Here we have briefly introduced the first stage of this development: tensor integral categories. The next stage, Cartesian integral categories, is actually well in hand. The coKleisli category of an integral category is a Cartesian integral category. Furthermore, Cartesian integral categories have a term logic which has a more "classic" feel: we borrowed parts of this term logic to help motivate this paper. The study of integration in restriction categories and tangent categories is, by comparison, in its earliest stages.

Acknowledgements. The authors would like to thank Rick Blute for drawing both authors' attention to Rota-Baxter algebras. Integral categories simply would not have developed so rapidly without this basic inspiration. Jonathan Gallagher reminded us of Ehrhard's work at exactly the right moment, while Kristine Bauer provided continual constructive criticism during the evolution of our thoughts.

References

1 Glen Baxter et al. An analytic problem whose solution follows from a simple algebraic identity. *Pacific J. Math*, 10(3):731–742, 1960.

2 R. Blute, J. R. B. Cockett, and R. A. G. Seely. Cartesian differential storage categories. *Theory and Applications of Categories*, 30(18):620–686, 2015.

3 R. F. Blute, J. Robin B. Cockett, and R. A. G. Seely. Cartesian differential categories. *Theory and Applications of Categories*, 22(23):622–672, 2009.

4 Richard Blute, J. R. B. Cockett, Timothy Porter, and R. A. G. Seely. Kähler categories. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 52(4):253–268, 2011.

5 Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *arXiv preprint arXiv:1006.3140*, 2010.

6 Richard Blute, Rory B. B. Lucyshyn-Wright, and Keith O'Neill. Derivations in codifferential categories. *arXiv preprint arXiv:1505.00220*, 2015.

7 Richard F. Blute, J. Robin B. Cockett, and Robert A. G. Seely. Differential categories. *Mathematical structures in computer science*, 16(06):1049–1083, 2006.

8 Raoul Bott and Loring W. Tu. *Differential forms in algebraic topology*, volume 82. Springer Science & Business Media, 2013.

9 J. Robin B. Cockett and Geoff S. H. Cruttwell. Differential Structure, Tangent Structure, and SDG. *Applied Categorical Structures*, 22(2):331–417, 2014.

10 J. R. B. Cockett, G. S. H. Cruttwell, and J. D. Gallagher. Differential restriction categories. *Theory and Applications of Categories*, 25(21):537–613, 2011.

11 Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Mathematical Structures in Computer Science*, pages 1–66, 2017.

12 Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, 2003.

13 Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006.

14 Marcelo P. Fiore. Differential structure in models of multiplicative biadditive intuitionistic linear logic. In *International Conference on Typed Lambda Calculi and Applications*, pages 163–177. Springer, 2007.

15 Li Guo. *An introduction to Rota-Baxter algebra*, volume 2. International Press Somerville, 2012.

16 André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991.

17 Serge Lang. Algebra revised third edition. *Graduate Texts in Mathematics*, 1(211):ALL–ALL, 2002.

18 J.-S. P. Lemay. *Integral Categories and Calculus Categories*. University of Calgary, 2017.

19 Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

20 Gian-Carlo Rota. Baxter algebras and combinatorial identities. I. *Bulletin of the American Mathematical Society*, 75(2):325–329, 1969.

21 Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.

22 Charles A. Weibel. *An introduction to homological algebra*. Cambridge university press, 1995.

Partial Elements and Recursion via Dominances in Univalent Type Theory

Martín H. Escardó[1] **and Cory M. Knapp**[2]

1 **School of Computer Science, University of Birmingham, Birmingham, UK**
`m.escardo@cs.bham.ac.uk`
2 **School of Computer Science, University of Birmingham, Birmingham, UK**
`cmk497@cs.bham.ac.uk`

Abstract

We begin by revisiting partiality in univalent type theory via the notion of dominance. We then perform first steps in constructive computability theory, discussing the consequences of working with computability as property or structure, without assuming countable choice or Markov's principle. A guiding question is what, if any, notion of partial function allows the proposition "all partial functions on natural numbers are Turing computable" to be consistent.

1998 ACM Subject Classification F.4.1 [Mathematical Logic] Lambda Calculus and Related Systems, F.1.1 [Models of Computation] Computability Theory

Keywords and phrases univalent type theory, homotopy type theory, partial function, dominance, recursion theory, computability theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.21

1 Introduction

We begin by revisiting partial functions in type theory [4, 3, 7, 5, 8, 21]. We work informally, but rigorously, in a univalent type theory as in the HoTT Book [23], with constructive content as developed in cubical type theory [9]. We then perform first steps in computability theory, where, in particular, we discuss the consequences of working with computability as property or as structure, in the absence of choice axioms, excluded middle and Markov's principle.

A guiding question is what, if any, notion of partial function allows the proposition "all partial functions from $\mathbb{N}$ to $\mathbb{N}$ are Turing computable" to be consistent in constructive univalent type theory. We begin with a natural, but too liberal, notion, that allows partial functions that are not computable in any reasonable sense to be easily constructed. We then use dominances [20] to try to restrict the notion of partial function (Section 2) so that this would be possible, discussing the difficulties that arise in this attempt (Section 3), and their relation to previous work by other authors (Section 4).

Dominances also occur in synthetic domain theory [12, 16, 24], synthetic computability theory [20, 1], and synthetic topology [20, 2]. In such synthetic approaches, one typically considers countable dependent choice and Markov's principle, in addition to axioms that contradict the principle of excluded middle. For example, in his synthetic approach to computability, Bauer [1] works with a non-classical axiom saying that there are enumerably many enumerable subsets of $\mathbb{N}$. We emphasize that, although we do use dominances, our approach is not synthetic, and, moreover, is compatible with classical logic.

In Section 2 we define a type $X \rightharpoonup Y$ of partial functions for any two types X and Y, so that partial functions correspond to ordinary (total) functions $A \to Y$ defined on a type A embedded into X, and show that it is equivalent to $X \to \mathcal{L}\,Y$ where $\mathcal{L}\,Y$ is a type of partial

elements of Y. The lifted type $\mathcal{L}\,Y$ is a directed-complete partially ordered set if Y is a set in the sense of univalent type theory, and this can be used to solve recursive definitions involving Scott continuous functions.

In Section 3, we define a Gödel encoding of Turing machines by natural numbers in the usual way, together with a Kleene-bracket function

$$\{-\} : \mathbb{N} \to (\mathbb{N} \to \mathcal{L}\,\mathbb{N}),$$

and define a function $\mathbb{N} \to \mathcal{L}\,\mathbb{N}$ to be computable if it is in the image of Kleene bracket. This can be formulated in two ways:

$$\begin{aligned} \text{computationalStructure}(f) &\stackrel{\text{def}}{=} \Sigma(e : \mathbb{N}), \{e\} = f, \\ \text{isComputable}(f) &\stackrel{\text{def}}{=} (\exists(e : \mathbb{N}), \{e\} = f) = \|\text{computationalStructure}(f)\|\,. \end{aligned}$$

The first definition says that a computational structure on f is a Gödel number of a Turing machine that computes f. The second definition says that f is computable if there is such a Gödel number, without exhibiting any. Here $\|-\|$ denotes the (propositional) truncation operator, which collapses a type to a subsingleton by identifying all its elements. Existence as a proposition is taken to be the truncation of existence as structure:

$$\exists(x : X), A(x) \stackrel{\text{def}}{=} \|\Sigma(x : X), A(x)\|\,.$$

We emphasize, for comparison, that in topos theory this is a theorem (often taken as a definition), expressing existence in terms of Σ, where $\|X\|$ is known as the *support* of the object X (the image of the unique map into the terminal object 1).

We can prove Rogers' version of Kleene's second recursion theorem, which says that for every computable total function $f : \mathbb{N} \to \mathbb{N}$, there is $e : \mathbb{N}$ with $\{e\} = \{f(e)\}$, in the following two versions:

$$\begin{aligned} &\Pi(f : \mathbb{N} \to \mathbb{N}), \text{computationalStructure}(f) \to \Sigma(e : \mathbb{N}), \{e\} = \{f(e)\}, \\ &\Pi(f : \mathbb{N} \to \mathbb{N}), \text{isComputable}(f) \to \exists(e : \mathbb{N}), \{e\} = \{f(e)\}, \end{aligned}$$

where the second version is a direct consequence of the first, by the universal property of propositional truncation.

For some statements it does matter whether computability is taken as property or structure. Consider, for example, the type

$$\Pi(f : \mathbb{N} \to \mathbb{N}), \text{computationalStructure}\, f$$

of functions that assign computational structure to every total function $\mathbb{N} \to \mathbb{N}$ (analogous to Lisp's quote operator). In the presence of function extensionality, the assumption that the above type is inhabited leads to a contradiction [22]. But the proposition

$$\Pi(f : \mathbb{N} \to \mathbb{N}), \text{isComputable}\, f$$

is consistent in a type theory with function extensionality and propositional truncation, with the effective topos as a model [11].

On the other hand, the statement that every *partial* function $\mathbb{N} \rightharpoonup \mathbb{N}$ is computable is false, even when the computational structure is not given. In fact, there is an easily defined partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ that semidecides the complement of the Halting Set, in the sense that $f(x)$ is defined iff $\{x\}(x)$ is undefined, and hence is not computable (Section 3.4).

This motivates the consideration of dominances, to restrict the allowed domains of definition of partial elements (Section 2). Of particular interest is the *Rosolini dominance* [20], which consists of the propositions of the form $\exists(n : \mathbb{N}), \alpha(n) = 1$ with $\alpha : \mathbb{N} \to 2$ (Section 2.4). This is related to various partiality monads considered in the literature, as discussed in the concluding Section 4.

2 Partial functions and recursion via dominances

2.1 Partial functions

Assuming univalence, the type of functions $X \to Y$ is equivalent to the type of functional relations [17], where $\mathcal{U}$ is a type universe:

$$(X \to Y) \simeq \Sigma(R : X \to Y \to \mathcal{U}), \Pi(x : X), \mathrm{isContr}\,(\Sigma(y : Y), R\,x\,y)\,. \tag{1}$$

Recall that a type A is called contractible (or a singleton), written $\mathrm{isContr}(A)$, if we have $a_0 : A$, called a center of contraction of A, which is equal to every element of A:

$$\mathrm{isContr}(A) \stackrel{\mathrm{def}}{=} \Sigma(a_0 : A), \Pi(a : A), a_0 = a.$$

The above use of contractibility can be read as saying that there is a unique (y, r), with $y : Y$ and $r : R\,x\,y$, rather than the weaker statement that there is a unique $y : Y$ satisfying $R\,x\,y$, with possibly several $r : R\,x\,y$ for this y.

A type A is a called a *proposition* (or said to have at most one element, or to be a subsingleton), written $\mathrm{isProp}(A)$, if any two of its elements are equal:

$$\mathrm{isProp}(A) \stackrel{\mathrm{def}}{=} \Pi(a, b : A), a = b.$$

Then in the following definition of a type of partial functions, we replace unique existence by the existence of at most one pair (y, r) with $y : Y$ and $r : R\,x\,y$:

$$\begin{aligned} (X \rightharpoonup Y) &\stackrel{\mathrm{def}}{=} \Sigma(R : X \to Y \to \mathcal{U}), \Pi(x : X), \mathrm{isProp}\,(\Sigma(y : Y), R\,x\,y) && (2)\\ &\simeq \Sigma(A : \mathcal{U})(e : A \to X), \mathrm{isEmbedding}(e) \times (A \to Y), && (3)\end{aligned}$$

where

$$\mathrm{isEmbedding}(e : A \to X) \stackrel{\mathrm{def}}{=} \Pi(x : X), \mathrm{isProp}(\Sigma(a : A), e(a) = x).$$

Then the type of ordinary functions $(X \to Y)$ is embedded into the type $(X \rightharpoonup Y)$ of partial functions, and we say that a partial function is total if it is in the image of this embedding.

In topos theory one has an isomorphism as in (1) with $\mathcal{U}$ replaced by the subobject classifier and the proposition $\mathrm{isContr}\,(\Sigma(y : Y), R\,x\,y)$ replaced by the unique existence of some $y : Y$ with $R\,x\,y$. This replacement is also possible in univalent type theory in the special case where the type Y is a set (meaning that for any two $y, y' : Y$, the identity type $y = y'$ is a subsingleton). For the general case, however, we need to keep $\mathcal{U}$ and work with the contractibility of the Σ type as above. Notice also that, in our type theory, the right-hand side of (1) is in a higher universe than the left-hand side (cf. Section 2.9).

2.2 Partial functions via lifting

If we define the lifting of a type Y (also referred to as the type of *partial elements* of Y) by

$$\begin{aligned} \mathcal{L}\,Y &\stackrel{\mathrm{def}}{=} 1 \rightharpoonup Y && (4)\\ &\simeq \Sigma(A : Y \to \mathcal{U}), \mathrm{isProp}\,(\Sigma(y : Y), A\,y) && (5)\\ &\simeq \Sigma(P : \mathcal{U}), \mathrm{isProp}\,P \times (P \to Y), && (6)\end{aligned}$$

then partial functions reduce to ordinary functions with lifted codomain:

$$(X \rightharpoonup Y) \simeq (X \to \mathcal{L}\,Y).$$

Working with the representation (6) of $\mathcal{L}\,Y$, we denote the first and third projections by

$$\text{extent} : \mathcal{L}\,Y \to \mathcal{U}, \qquad \text{value} : \Pi(u : \mathcal{L}\,Y), \text{extent}\,u \to Y. \tag{7}$$

Then $\mathcal{L}$ has a strong-monad structure, which we present as a Kleisli triple. The unit $\eta_X : X \to \mathcal{L}\,X$ is given by $\eta_X(x) \stackrel{\text{def}}{=} (1, -, \lambda p.x)$, where we denote a suppressed witness by "$-$". Given $f : X \to \mathcal{L}\,Y$, we define its Kleisli extension $f^\sharp : \mathcal{L}\,X \to \mathcal{L}\,Y$ by

$$f^\sharp(P : \mathcal{U}, -, \phi : P \to X) \stackrel{\text{def}}{=} (Q : \mathcal{U}, -, \gamma : Q \to Y),$$

$$\text{where} \quad Q \stackrel{\text{def}}{=} \Sigma(p : P), \text{extent}(f(\phi(p)) \quad \text{and} \quad \gamma(p, e) \stackrel{\text{def}}{=} \text{value}(f(\phi(p)))(e).$$

Then Kleisli composition corresponds to relational composition, so that post-composition with η_Y is an embedding of $(X \to Y)$ into $(X \to \mathcal{L}\,Y)$, often taken as an implicit coercion.

We say that an element of $\mathcal{L}\,Y$ is *defined* (or *total*) if it is in the image of $\eta_Y : Y \to \mathcal{L}\,Y$, which is equivalent to saying that its extent of definition is inhabited (and hence contractible). The type $\mathcal{L}\,Y$ has an *undefined* element $\bot$, with empty extent of definition. The assertion that every partial element is defined or else undefined is equivalent to the principle of excluded middle (every proposition is inhabited or empty).

2.3 Recursive definitions

If the type Y is a set, then $\mathcal{L}\,Y$ is a directed-complete partially ordered set with least element $\bot$. Define $- \leq - : \mathcal{L}\,Y \to \mathcal{L}\,Y \to \mathcal{U}$ by

$$(u \leq v) \stackrel{\text{def}}{=} \Sigma(t : \text{extent}(u) \to \text{extent}(v)), \Pi(p : \text{extent}(u)), \text{value}(u)(p) = \text{value}(v)(t(p)).$$

As Y is a set, this relation is proposition-valued, and reflexive, transitive and antisymmetric.

▶ **Theorem 1.** *If Y is a set, then $(\mathcal{L}\,Y, \leq)$ is a directed-complete poset.*

Proof. Given a directed family $u_i : \mathcal{L}\,Y$, we construct its join $u_\infty : \mathcal{L}\,Y$ as follows. Firstly, we let $\text{extent}(u_\infty) \stackrel{\text{def}}{=} \|\Sigma_i, \text{extent}(u_i)\|$, so that, by construction, we have $\text{isProp}(\text{extent}(u_\infty))$. To define $\text{value}(u_\infty) : \text{extent}(u_\infty) \to Y$, we first define a function $\phi : (\Sigma_i, \text{extent}(u_i)) \to Y$ by $\phi(i, p) = \text{value}(u_i)(p)$. By the directedness of the family, we see that ϕ is constant, in the sense that any two of its values are equal, and hence, by the assumption that Y is a set, ϕ factors through a unique $\|\Sigma_i, \text{extent}(u_i)\| \to Y$ by [14, Theorem 5.4], which we take as $\text{value}(u_\infty)$. We omit the routine verification that u_∞ is the least upper bound of the family u_i. ◀

This can be used to construct least fixed points of continuous functions in the usual way, and hence solve recursive functional equations, using the fact that a function type $X \to \mathcal{L}\,Y$ is also directed complete under the pointwise order.

2.4 Partial elements with semidecidable extent of definition

To try to make the statement that partial functions $\mathbb{N} \to \mathcal{L}\,\mathbb{N}$ are computable consistent, we need to restrict the supply of propositions that are allowed to arise as extents of definition of partial elements, motivating the subject of dominances, mentioned above and to be discussed shortly. The propositions that can arise as the extents of definition of partial elements from Turing machines are characterized as those of the form

$$\langle\alpha\rangle \stackrel{\text{def}}{=} \Sigma(n : \mathbb{N}), \alpha(n) = 1,$$

with $\alpha : \mathbb{N} \to 2$ computable and $\mathrm{isProp}\langle\alpha\rangle$ (Section 3.5). Rosolini considered partial elements with extents of the above form [20], without assuming f computable, which we refer to as *Rosolini propositions*:

$$\begin{aligned} \mathrm{isRosolini}(A) &\stackrel{\mathrm{def}}{=} \exists(\alpha : \mathbb{N} \to 2), \mathrm{isProp}\langle\alpha\rangle \times (A = \langle\alpha\rangle). \\ \mathrm{isSemiDecidable}(A) &\stackrel{\mathrm{def}}{=} \exists(\alpha : \mathbb{N} \to 2), \mathrm{isComputable}(\alpha) \times \mathrm{isProp}\langle\alpha\rangle \times (A = \langle\alpha\rangle). \end{aligned}$$

Assuming the (consistent, internal) statement that all (total) functions $\mathbb{N} \to \mathbb{N}$ are computable, the Rosolini and semidecidable propositions coincide, of course.

2.5 Dominances

A dominance is a set of propositions closed under Σ, and having the proposition 1 as a member. It is convenient to present a dominance by the following data and properties:

D1. A function $d : \mathcal{U} \to \mathcal{U}$.
Then a type X is called *dominant* if $d(X)$ holds.

D2. A function $\Pi(X : \mathcal{U}), \mathrm{isProp}(d(X))$.
(Being dominant is a proposition.)

D3. A function $\Pi(X : \mathcal{U}), d(X) \to \mathrm{isProp}(X)$.
(Dominant types are propositions.)

D4. An element of $d(1)$ where 1 is the terminal type.
(The proposition 1 is dominant.)

D5. A function $\Pi(P : \mathcal{U})(Q : P \to \mathcal{U}), d(P) \to (\Pi(p : P), d(Q(p))) \to d\,(\Sigma(p : P), Q(p))$.
(Dominant types are closed under Σ.)

The required unnamed element and functions (D2)–(D5) are unique as their types are propositions, and hence constitute a property of d. Notice that $d \stackrel{\mathrm{def}}{=} \mathrm{isProp}$ satisfies the dominance axioms, that is, the set of all propositions forms a dominance. In the general case, (D5) is equivalent to the seemingly weaker condition

$$\Pi(P, Q' : \mathcal{U}), d(P) \to (P \to d(Q')) \to d(P \times Q'),$$

which corresponds to the original definition of dominance given by Rosolini. (In one direction, consider the constant family $Q(p) \stackrel{\mathrm{def}}{=} Q'$. In the other, consider the type $Q' \stackrel{\mathrm{def}}{=} \Sigma(p : P), Q(p)$.)

2.6 Partial functions induced by a dominance

Given a dominance d, we define the type of d-partial functions by

$$(X \rightharpoonup_d Y) \stackrel{\mathrm{def}}{=} \Sigma(R : X \to Y \to \mathcal{U}), \Pi(x : X), d\,(\Sigma(y : Y), R\,x\,y)\,. \tag{8}$$

The axiom that the proposition 1 is dominant ensures that the identity relation is a d-partial function, and the axiom that dominant propositions are closed under Σ ensures that d-partial functions are closed under composition. In fact, if we have relations $R : X \to Y \to \mathcal{U}$ and $S : Y \to Z \to \mathcal{U}$, their composition is

$$(R; S)\,x\,z \stackrel{\mathrm{def}}{=} \Sigma(y : Y), R\,x\,y \times S\,y\,z,$$

and the closure of d under Σ ensures that the assumptions $\Pi(x : X), d\,(\Sigma(y : Y), R\,x\,y)$ and $\Pi(y : Y), d\,(\Sigma(z : Z), S\,y\,z)$ on R and S together imply that their composite $R; S$ satisfies the required condition $\Pi(x : X), d\,(\Sigma(z : Z), (R; S)\,x\,z)$. Indeed, we have that the type $\Sigma(z : Z), (R; S)\,x\,z$ is equivalent to $\Sigma((y, r) : \Sigma(y : Y), R\,x\,y), \Sigma(z : Z), S\,y\,z$ by the reshuffling map $(z, (y, (r, s))) \mapsto ((y, r), (z, s))$. Being the sum of the dominant propositions

$\Sigma(z : Z), S\,y\,z$ indexed by a dominant proposition $\Sigma(y : Y), R\,x\,y$, this type is itself a dominant proposition, as required. Conversely, by constructing suitable relations R and S depending on a given family $Q : P \to \mathcal{U}$ of dominant propositions indexed by a dominant proposition P, we can see that closure under Σ is necessary.

2.7 Partial elements induced by a dominance

The d-lifting, or type of d-partial elements of Y, is defined by

$$\begin{aligned} \mathcal{L}_d\,Y &\stackrel{\text{def}}{=} (1 \rightharpoonup_d Y) && (9)\\ &\simeq (\Sigma(A : Y \to \mathcal{U}), d\,(\Sigma(y : Y), A\,y)) && (10)\\ &\simeq (\Sigma(P : \mathcal{U}), d(P) \times (P \to X)) && (11) \end{aligned}$$

so that

$$(X \rightharpoonup_d Y) \simeq (X \to \mathcal{L}_d\,Y).$$

The dominance axioms ensure that the monad structure on $\mathcal{L}$ discussed above restricts to $\mathcal{L}_d$, again with Kleisli composition corresponding to relational composition. Trivial examples of dominances are

$$d_1 \stackrel{\text{def}}{=} \text{isContr}, \quad d_2(X) \stackrel{\text{def}}{=} (X = 0) + (X = 1), \quad d_\Omega \stackrel{\text{def}}{=} \text{isProp},$$

with total spaces 1, 2, and Ω, where $2 \simeq 1 + 1$ and $\Omega \stackrel{\text{def}}{=} \Sigma(P : \mathcal{U}), \text{isProp}\,P$, which satisfy

$$\mathcal{L}_{d_1}(X) \simeq X, \quad \mathcal{L}_{d_2}(X) \simeq X + 1\ , \quad \mathcal{L}_{d_\Omega}(X) = \mathcal{L}\,X.$$

(And hence $\mathcal{L}\,X \simeq X + 1$ iff the canonical map $2 \to \Omega$ is an equivalence iff excluded middle holds.)

2.8 The Rosolini dominance and choice

A dominance often considered in topos theory, typically in realizability toposes, is the *Rosolini dominance* consisting of the Rosolini propositions defined above [20, 12]. Then the statement in the internal language that "all partial functions $\mathbb{N} \rightharpoonup \mathbb{N}$ whose values have Rosolini extents of definition are computable" is valid in the effective topos [11, 20] and hence consistent in a dependent type theory with function extensionality, propositional truncations *and some amount of choice*. Countable choice, which holds in the effective topos assuming a classical meta-theory for its study, is used to prove that the Rosolini propositions form a dominance [20, Chapter 5.2].

We work with the axiom of choice in the form "a product of inhabited sets is inhabited" [23]:

$$(\Pi(x : X), \|Y(x)\|) \to \|\Pi(x : X), Y(x)\|\,,$$

for a set $X : \mathcal{U}$ and a family of sets $Y : X \to \mathcal{U}$. When the set X is the type of natural numbers, this is *countable choice*. When X ranges over propositions, this is *propositional choice*, which is shown in [14] to be equivalent to

$$(P \to \|A\|) \to \|P \to A\|\,,$$

where P ranges over propositions and A over sets. Notice that propositional choice holds for decidable P, and hence excluded middle implies propositional choice. But the decidability of Rosolini propositions is known as LPO (the limited principle of omniscience), and is not constructively provable in any variety of constructive mathematics [6]. However, it is known that excluded middle doesn't imply countable choice, at least not in topos logic.

▶ **Theorem 2.** *For any set A, the following are equivalent:*
1. *Countable choice for families $Y(n) \stackrel{def}{=} (\alpha(n) = 1) \to A$ with $\alpha : \mathbb{N} \to 2$.*
2. *Propositional choice from Rosolini propositions to A.*

Proof. Consider the following propositions:
1. $\langle\alpha\rangle \to \|A\|$,
2. $(\Sigma(n : \mathbb{N}), \alpha(n) = 1) \to \|A\|$,
3. $\Pi(n : \mathbb{N}), ((\alpha(n) = 1) \to \|A\|)$,
4. $\Pi(n : \mathbb{N}), \|(\alpha(n) = 1) \to A\|$,
5. $\|\Pi(n : \mathbb{N}), (\alpha(n) = 1) \to A\|$,
6. $\|(\Sigma(n : \mathbb{N}), \alpha(n) = 1) \to A\|$,
7. $\|\langle\alpha\rangle \to A\|$.

The implication (4) → (5) is the above instance of countable choice, and the implication (1) → (7) is the above instance of propositional choice. Hence the chain of implications (1) → (2) → (3) → (4) → (5) → (6) → (7) gives propositional choice from countable choice, and the chain of implications (4) → (3) → (2) → (1) → (7) → (6) → (5) gives countable choice from propositional choice. ◀

Hence this particular case of countable choice is implied by excluded middle, and in fact already by LPO, because propositional choice for Rosolini propositions is. To show that the Rosolini propositions form a dominance, it is necessary and sufficient to have choice from Rosolini propositions P to *Rosolini structures* A, namely types of the form

$$\text{RosoliniStructure}(Q) \stackrel{\text{def}}{=} \Sigma(\alpha : \mathbb{N} \to 2), \text{isProp}\langle\alpha\rangle \times (Q = \langle\alpha\rangle)$$

with $Q : \mathcal{U}$. Notice that the type of Rosolini structures is a subtype of the *one-point compactification of* $\mathbb{N}$, namely the type $\mathbb{N}_\infty \stackrel{\text{def}}{=} \Sigma(\alpha : \mathbb{N} \to 2), \text{isProp}\langle\alpha\rangle$.

▶ **Theorem 3.** *The following are equivalent:*
1. *Choice from Rosolini propositions to Rosolini structures.*
2. *The Rosolini propositions form a dominance.*

Proof. (⇓): It suffices to show that, for any $\alpha : \mathbb{N} \to 2$ with $\text{isProp}\langle\alpha\rangle$ and $Q : \mathcal{U}$, we have $(\langle\alpha\rangle \to \text{isRosolini}\, Q) \to \text{isRosolini}(\langle\alpha\rangle \times Q)$, which amounts, by propositional univalence, to

$$(\langle\alpha\rangle \to \|\Sigma(\beta : \mathbb{N} \to 2), \text{isProp}\langle\beta\rangle \times (Q = \langle\beta\rangle)\|) \to \|\Sigma(\gamma : \mathbb{N} \to 2), (\langle\alpha\rangle \times Q) \iff \langle\gamma\rangle\|\,,$$

and hence, by Rosolini propositional choice, to

$$\|\langle\alpha\rangle \to \Sigma(\beta : \mathbb{N} \to 2), \text{isProp}\langle\beta\rangle \times (Q = \langle\beta\rangle)\| \to \|\Sigma(\gamma : \mathbb{N} \to 2), (\langle\alpha\rangle \times Q) \iff \langle\gamma\rangle\|\,,$$

and so, because truncation is functorial, it suffices to show that

$$(\langle\alpha\rangle \to \Sigma(\beta : \mathbb{N} \to 2), \text{isProp}\langle\beta\rangle \times (Q = \langle\beta\rangle)) \to \Sigma(\gamma : \mathbb{N} \to 2), (\langle\alpha\rangle \times Q) \iff \langle\gamma\rangle.$$

Assume an element ϕ of the premise, and define $\beta : \langle\alpha\rangle \to (\mathbb{N} \to 2)$ by $\beta(i, p) = \text{pr}_1(\phi(i, p))$. Then define $\gamma(n) = 1 \iff \Sigma(i, j \leq n), ((i = n) + (j = n)) \times \Sigma(p : \alpha(i) = 1), \beta(i, p)(j) = 1$. We omit the verification that $\text{isProp}\langle\gamma\rangle$ and that $(\langle\alpha\rangle \times Q) \iff \langle\gamma\rangle$.

(⇑): Assume $P \to \|\text{RosoliniStructure}\, Q\|$ for some Rosolini proposition P and some proposition Q, that is, $P \to \text{isRosolini}\, Q$. By the dominance axiom, $\text{isRosolini}(P \times Q)$. Because $P \to ((P \times Q) = Q)$, by propositional univalence, we have

$$\text{RosoliniStructure}(P \times Q) \to (P \to \text{RosoliniStructure}(Q)).$$

By functoriality of truncation, this gives $\text{isRosolini}(P \times Q) \to \|P \to \text{RosoliniStructure}(Q)\|$. Because the premise holds, $\|P \to \text{RosoliniStructure}(Q)\|$, as required. ◀

But it doesn't seem to be possible to replace countable choice by its weakening discussed above in Theorem 4 and Corollary 5 below. If we order propositions by $(P \leq Q) \stackrel{\text{def}}{=} (P \to Q)$, then any family of propositions has a least upper bound, given by the truncation of its sum.

▶ **Theorem 4.** *Assuming countable choice, the Rosolini propositions are closed under countable joins in the complete lattice of all propositions.*

Proof. We have to show that $\|\Sigma_i, P_i\|$ is a Rosolini proposition for any countable family P_i of Rosolini propositions. Countable choice gives

$$\|\Sigma(\alpha : \mathbb{N} \times \mathbb{N} \to 2), \Pi(i : \mathbb{N}), \mathrm{isProp}\langle\alpha(i,-)\rangle \times (P_i = \langle\alpha(i,-)\rangle)\|\,.$$

The last equation amounts to $P_i = (\Sigma_j, \alpha(i,j) = 1)$, which gives $(\Sigma_i, P_i) = (\Sigma_{i,j}, \alpha(i,j) = 1)$ and hence $\|\Sigma_i, P_i\| = \|\Sigma_{i,j}, \alpha(i,j) = 1\|$. Now if we define $\beta(n) = 1$ iff there is a minimal pair $i, j \leq n$ in the lexicographic order with $\alpha(i,j) = 1$, then β satisfies $\mathrm{isProp}\langle\beta\rangle$ and $\langle\beta\rangle \iff \|\Sigma_{i,j}, \alpha(i,j) = 1\|$, which shows that $\|\Sigma_i, P_i\|$ is a Rosolini proposition, as required. ◀

▶ **Corollary 5.** *If Y is a set, then countable choice implies that the Rosolini lifting of Y has joins of ascending sequences.*

Proof. The construction is as that of Theorem 1, with the additional step of showing that $\|\Sigma_i, \mathrm{extent}(u_i)\|$ is a Rosolini proposition, which is given by Theorem 4. ◀

However, Coquand, Mannaa, and Ruch [10] have shown that countable choice cannot be proved in dependent type theory with one univalent universe and propositional truncation, and discussions in research mailing lists for homotopy type theory and constructive mathematics with them and with Andrew Swan indicate that their negative result also applies to our weakening of countable choice.

2.9 Size matters

We now discuss the universe levels of the above constructions (which we have checked using Agda). We assume we have a function $d : \mathcal{U}_0 \to \mathcal{U}_0$, for example $d = \mathrm{isProp}$ or $d = \mathrm{isRosolini}$. We decorate the above constructions with universe levels j, k, l as superscripts. For a type $Y : \mathcal{U}_j$, we have $\mathcal{L}_d(Y) : \mathcal{U}_{\max(j,1)}$, so that lifting has the following type:

$$\mathcal{L}_d^j : \mathcal{U}_j \to \mathcal{U}_{\max(j,1)}.$$

In particular, we have $\mathcal{L}^0 : \mathcal{U}_0 \to \mathcal{U}_1$, and $\mathcal{L}^1 : \mathcal{U}_1 \to \mathcal{U}_1$, and $\mathcal{L}^{j+1} : \mathcal{U}_{j+1} \to U_{j+1}$, where we have elided the subscript d to avoid notational clutter. That is, lifting raises the universe level only at the lowest universe. With propositional resizing [25], assuming $\mathrm{isProp}(d(P))$, we would get that $\mathcal{L}^0 : \mathcal{U}_0 \to \mathcal{U}_0$ and hence $\mathcal{L}^j : \mathcal{U}_j \to U_j$ for every j so that universe levels are not raised. But resizing is not needed for our purposes, if we are willing to work with higher universes, and $\mathcal{U}_1$ suffices for our discussion of computation of partial functions in Section 3.

For the unit we have the type $\eta^j : \Pi(X : \mathcal{U}_j), X \to \mathcal{L}^j(X)$. Assuming that $d : \mathcal{U}_0 \to \mathcal{U}_0$ is closed under Σ in the sense discussed above, we get an extension operator $(-)^\sharp$ of type

$$\Pi(X : \mathcal{U}_j, Y : \mathcal{U}_k), (X \to \mathcal{L}^k(Y)) \to (\mathcal{L}^j(X) \to \mathcal{L}^k(Y)),$$

which for $j = k = 0$ specializes to $\Pi(X, Y : \mathcal{U}_0), (X \to \mathcal{L}^0(Y)) \to (\mathcal{L}^0(X) \to \mathcal{L}^0(Y))$. From this we get a Kleisli composition operator of type

$$\Pi(X : \mathcal{U}_j)(Y : \mathcal{U}_k)(Z : \mathcal{U}_l), (Y \to \mathcal{L}^l(Z)) \to (X \to \mathcal{L}^k(Y)) \to (X \to \mathcal{L}^l(Z)),$$

which specializes to $\Pi(X, Y, Z : \mathcal{U}_0), (Y \to \mathcal{L}^0(Z)) \to (X \to \mathcal{L}^0(Y)) \to (X \to \mathcal{L}^0(Z))$. With the general universe levels, the equations for Kleisli triples can be written down (they type check) and proved. For the multiplication induced by the extension operator, we have the type

$$\mu^j : \Pi(X : \mathcal{U}_j), \mathcal{L}^{\max(j,1)}(\mathcal{L}^j(X)) \to \mathcal{L}^j(X)$$

which specializes to $\Pi(X : \mathcal{U}_0), \mathcal{L}^1(\mathcal{L}^0(X)) \to \mathcal{L}^0(X)$ and $\Pi(X : \mathcal{U}_1), \mathcal{L}^1(\mathcal{L}^1(X)) \to \mathcal{L}^1(X)$. Again the monad laws, decorated with general universe levels, can be written down and proved. Of course, we don't have a monad because the universe levels give the above non-standard types for the unit and multiplication. Notice also that, because types don't form a category, but something like an ∞-category, a more refined notion of monad would be needed, even ignoring the issue with universe levels. In any case, our unrefined version is enough to get partial functions $(-) \to \mathcal{L}(-)$ with a natural Kleisli composition operation induced by the monad-like structure.

We observe that by the results of [18], if our type theory includes *graph quotients*, then the total space of $d \stackrel{\text{def}}{=} \text{isRosolini}$ is small, as it is the image of a function $\mathbb{N}_\infty \to \mathcal{U}_0$ into the locally small type $\mathcal{U}_0$, so that lifting, extension, unit and multiplication preserve universe levels even at the lowest universe.

3 Computable functions

We now demonstrate a way to develop computability theory constructively, and then in Section 3.5 relate this to our previous discussion of partial functions and dominances. A main point of this redevelopment is to make sure we don't use countable choice or Markov's principle, in addition to non-constructive principles. While Turing machines provide an intuitively convincing notion of computation and are easy to define in a constructive setting, they are cumbersome to work with directly. We will instead work with a model which is a modest abstraction of the notion of Turing machine.

3.1 Primitive recursive functions

Our model will make use of (unary) primitive recursive functions, so we quickly recall a few basic facts about primitive recursive functions. The classical presentation of the results mentioned here (such as that in [13], [19], or [15]) can be immediately adapted to fit in univalent type theory. The key point is that the functions used are all computable and total, so there's no place where non-constructive notions are likely to appear.

▶ **Definition 6.** The type family $\text{PR} : \mathbb{N} \to \mathcal{U}$ of *primitive recursive combinators (of arity n)* is defined inductively by

1. $\text{s} : \text{PR}_1$;
2. for any $n : \mathbb{N}$ and $k < n$ we have $\text{p}^n_k : \text{PR}_n$;
3. for any $n, k : \mathbb{N}$ we have $\text{c}^n_k : \text{PR}_n$;
4. if $f : \text{PR}_n$ and $g_i : \text{PR}_m$ for each $0 < i \leq n$, then $f\langle g_1, \ldots, g_n\rangle : \text{PR}_m$.
5. if $f : \text{PR}_{n+2}$ and $g : \text{PR}_n$, then $\text{r}_{f,g} : \text{PR}_{n+1}$.

We can define a function $\text{eval}_n : \text{PR}_n \to \mathbb{N}^n \to \mathbb{N}$ in the obvious way, such that $\text{eval}(t)$ is primitive recursive (in the usual sense) for each PR term t. We write $\text{isPR}_n(f)$ if $f : \mathbb{N}^n \to \mathbb{N}$ is in the image of eval, and say that f is primitive recursive if $\text{isPR}(f)$.

▶ **Theorem 7** (Enumeration Theorem for Primitive Recursive functions)**.** *For each $n : \mathbb{N}$, there is an injective function $e : \mathrm{PR}_n \to \mathbb{N}$, and a function $\{-\} : \mathbb{N} \to (\mathbb{N}^n \to \mathbb{N})$ such that for all $f : \mathrm{PR}_n$ and $x : \mathbb{N}^n$,*

$$\{e_f\}(x) = \mathrm{eval}(f, x).$$

A *primitive recursive characteristic function* for a predicate $R : \mathbb{N}^n \to \Omega$ is a primitive recursive function $\chi_R : \mathbb{N} \to \mathbb{N}$ such that for all $x : \mathbb{N}^n$

$$\chi_R(x) = 1 \Leftrightarrow R(x) \wedge \chi_R(x) = 0 \Leftrightarrow \neg R(x).$$

We will write $\mathrm{PR}(R)$ for the type of p.r. characteristic functions of R, and say that R is *primitive recursive* if $\mathrm{PR}(R)$. As $\mathrm{isPR}(f)$ is a proposition for any $f : \mathbb{N}^n \to \mathbb{N}$, so is $\mathrm{PR}(R)$. Observe that if R has a primitive recursive characteristic function, then R is complemented.

Many of our arguments will make use of functions with arity greater than 1, but we only use unary primitive recursive functions in what follows, so we will make use of primitive recursive pairing functions $\langle - \rangle : \mathbb{N}^n \to \mathbb{N}$, with primitive recursive projections. We will tacitly use the fact that if $f : \mathbb{N}^n \to \mathbb{N}$ is primitive recursive, then there is a primitive recursive function $\hat{f} : \mathbb{N} \to \mathbb{N}$ such that

$$\hat{f}\langle x_1, \ldots, x_n \rangle = f(x_1, \ldots, x_n).$$

Similarly, we will treat functions $\mathbb{N} + \mathbb{N} \to \mathbb{N}$ as functions $\mathbb{N} \to \mathbb{N}$, using a fixed bijection $\mathbb{N} + \mathbb{N} \simeq \mathbb{N}$ which makes the encodings of the inclusions inl and inr primitive recursive.

3.2 Abstract Turing machines

Our model abstracts away the details of the initialization and transition functions of a Turing machine, giving us the following definition.

▶ **Definition 8.** An *abstract Turing machine* is a pair (i, s) of functions $i : \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N} + \mathbb{N}$. The function i is called the *initialization function* and s is called the *transition function*. A *primitive recursive machine*, or recursive machine for short, is a pair $(i, s) : \mathrm{PR}_1 \times \mathrm{PR}_1$.

ATM is the type of abstract Turing machines, and RM is the type of recursive machines. We will treat recursive machines as ATMs by implicitly using the map defined by evaluating the combinators and composing along our fixed bijection $\mathbb{N} \to \mathbb{N} + \mathbb{N}$.

We can evaluate abstract Turing machines via a function $\mathrm{eval} : \mathrm{ATM} \to (\mathbb{N} \to \mathcal{L}\,\mathbb{N})$ as follows: Given $(i, s) : \mathrm{ATM}$, let $s' : \mathbb{N} + \mathbb{N} \to \mathbb{N} + \mathbb{N}$ be the function

$$s'(\mathrm{inl}\, x) = s(x),$$
$$s'(\mathrm{inr}\, y) = \mathrm{inr}\, y.$$

Now define for each $k : \mathbb{N}$ the function $\mathrm{run}_k : \mathrm{ATM} \to \mathbb{N} \to (\mathbb{N} + \mathbb{N})$ by

$$\mathrm{run}_k(m, x) \stackrel{\mathrm{def}}{=} s'^k(\mathrm{inl}(i(x)))$$

And so we have for fixed m,

$$R_m(x, y) \stackrel{\mathrm{def}}{=} \exists(k : \mathbb{N}), \mathrm{run}_k(m, x) = \mathrm{inr}\, y.$$

Then $\mathrm{eval}(m)$ is the partial function with R_m as its graph.

The partial function $\mathrm{eval}(m)$ is the function *computed by* m. We will sometimes abuse notation and write $m(x)$ for $\mathrm{eval}(m, x)$. For a partial function f, we let $\mathrm{computationalStructure}(f)$ be the type of recursive machines computing f, and $\mathrm{isComputable}(f)$ be the propositional truncation of $\mathrm{computationalStructure}(f)$. We let Comp be the type of computable functions:

$$\mathrm{Comp} \stackrel{\mathrm{def}}{=} \Sigma(f : \mathbb{N} \rightharpoonup \mathbb{N}), \mathrm{isComputable}(f).$$

The distinction between $\mathrm{computationalStructure}(f)$ and $\mathrm{isComputable}(f)$ is necessary. By the enumeration theorem for primitive recursive functions, we have a map

$$(\Sigma(f : \mathbb{N} \rightharpoonup \mathbb{N}), \mathrm{computationalStructure}(f)) \to \mathbb{N},$$

given by $(i, s) \mapsto \langle e_i, e_s \rangle$, which is easily seen to be an embedding.

On the other hand, if there is an embedding $F : \mathrm{Comp} \to \mathbb{N}$, then equality in Comp is complemented, since we have for $f, g : \mathrm{Comp}$ that $f = g \simeq F(f) = F(g)$, and equality between naturals is complemented. In particular, we would have a way to determine whether a (computable) function is equal to $\lambda x.0$, so we would have WLPO for computable functions. This means that we also cannot even expect to have an embedding

$$\mathrm{Comp} \to \Sigma(f : \mathbb{N} \rightharpoonup \mathbb{N}), \mathrm{computationalStructure}(f).$$

That is, we have no way of finding a program for a function just by knowing one exists.

As Turing machines can be encoded so that the initialization and transition functions are primitive recursive, every Turing machine is a recursive machine; conversely, given primitive recursive functions i and s, we can construct a Turing machine that runs i, and then iteratively runs s.

A simple programming exercise shows that if m and n are abstract Turing machines computing f and g respectively, then there is a machine $m; n$ computing $g \circ f$. Moreover, it is easy to see that when m and n are recursive machines, then so is $m; n$. Using the universal property of truncations this gives us,

$$\Pi(f, g : \mathbb{N} \rightharpoonup \mathbb{N}), \mathrm{isComputable}(f) \to \mathrm{isComputable}(g) \to \mathrm{isComputable}(g \circ f).$$

We have a minimization operator for partial functions, $\mu : (\mathbb{N} \to \mathcal{L}\,\mathbb{N}) \to \mathcal{L}\,\mathbb{N}$, with

$$\mathrm{extent}(\mu f) \stackrel{\mathrm{def}}{=} \Sigma(k : \mathbb{N}), f(k) = \eta 0 \times \Pi(j < k), \Sigma(n : \mathbb{N}), f(j) = \eta(n + 1),$$

and $\mathrm{value}(\mu f) \stackrel{\mathrm{def}}{=} \mathrm{pr}_1$. More generally, we may define the minimization $\mu y.R(y) : \mathcal{L}\,\mathbb{N}$ of a (decidable) predicate $R : \mathbb{N} \to \Omega$

$$\mathrm{extent}(\mu y.R(y)) \stackrel{\mathrm{def}}{=} \Sigma(y : \mathbb{N}), R(y) \times \Pi(x < y), \neg R(x).$$

and $\mathrm{value}(\mu y.R(y)) \stackrel{\mathrm{def}}{=} \mathrm{pr}_1$. When $R : \mathbb{N}^2 \to \Omega$ is primitive recursive, it is easy to see that $\mathrm{isComputable}(\lambda x.\mu y.R(x, y))$, by checking $R(x, n)$ for each n. Via dovetailing (by using run_k instead of eval), we can make this work with any *recursive* predicate, as expected from classical recursion theory.

3.3 Basic Recursion Theory

We are now able to show how to develop basic recursion theory via recursive machines. We roughly follow the presentation in Odifreddi [15]; in fact, the proofs given by Odifreddi for the results stated here translate to our framework with only minor adjustment. We begin with Kleene's Normal Form theorem.

▶ **Theorem 9** (Kleene's Normal Form Theorem). *There is an injection* $e : \mathrm{RM} \to \mathbb{N}$*, a primitive recursive predicate* T*, and a primitive recursive function* U *such that for any recursive machine* m *and* $x : \mathbb{N}$ *we have*

$$\mathrm{eval}(m, x) = (\Sigma(y : \mathbb{N}), \mathrm{T}(e_m, x, y)\ ,\ -\ ,\ \mathrm{U} \circ \mathrm{pr}_1)\,.$$

This says that $\mathrm{eval}(m, x)$ is defined iff there is a y such that $\mathrm{T}(e_m, x, y)$, and that when there is such a y, the value of $\mathrm{eval}(m, x)$ is $\mathrm{U}(y)$. As for any $e, x : \mathbb{N}$ there is a unique y such that $\mathrm{T}(e, x, y)$, we actually have that $\Sigma(y : \mathbb{N}), \mathrm{T}(e_m, x, y)$ is the extent of definition of $\mu y.\, \mathrm{T}(e_m, x, y)$, and we get the usual statement of the Normal Form Theorem.

▶ **Corollary 10.** *For any recursive machine* m *and* $x : \mathbb{N}$*, we have*

$$\mathrm{eval}(m, x) = \mathcal{L}\,\mathrm{U}(\mu y.\, \mathrm{T}(e_m, x, y)).$$

This result gives us the Kleene-bracket function $\{-\} : \mathbb{N} \to (\mathbb{N} \to \mathcal{L}\,\mathbb{N})$, defined by

$$\{e\}(x) \stackrel{\mathrm{def}}{=} \mathcal{L}\,\mathrm{U}(\mu y.\, \mathrm{T}(e, x, y)).$$

In order to be able to apply $\{-\}$ to a partial natural number, let $\{-\}^* : \mathcal{L}\,\mathbb{N} \to (\mathbb{N} \to \mathcal{L}\,\mathbb{N})$ be the function obtained by Kleisli extension in the first component.

Rogers' Fixed Point Theorem and Kleene's Second Recursion Theorem can now be proved in the standard way; observe, however, that the precise statement of Rogers' result must be modified to make sense in our setting, since the partial functions obtained from recursive machines (even total ones) have type $\mathbb{N} \to \mathcal{L}\,\mathbb{N}$. The proofs will require the S_n^m Theorem:

▶ **Theorem 11** (S_n^m Theorem). *There is a primitive recursive* $s : \mathbb{N}^2 \to \mathbb{N}$ *such that for all* $e, x, y : \mathbb{N}$ *we have* $\{s(e, x)\}y = \{e\}\langle x, y\rangle$*.*

▶ **Theorem 12** (Rogers' fixed point theorem). *For any recursive machine* m *such that* $\mathrm{eval}(m)$ *is total,* $\Sigma(n : \mathbb{N}), \{n\} = \{m(n)\}^*$*.*

Proof. Let $g : \mathbb{N} \to \mathbb{N}$ be defined by

$$g\langle x, y\rangle \stackrel{\mathrm{def}}{=} \{\{x\}(x)\}^*(y).$$

As this has a recursive machine, we have a code e_g. Defined $h(x) = s(e_g, x)$. Now let $e = e_{\mathrm{eval}(m)\circ h}$ and $n = h(e)$. Then we have

$$\{n\}y = \{h(e)\}y = \{s(e_g, e)\} = \{e_g\}\langle e, y\rangle = \{\{e\}(e)\}^*(y) = \{m(he)\}^*(y) = \{m(n)\}^*(y).\blacktriangleleft$$

▶ **Theorem 13** (Kleene's Second Recursion Theorem). *For any recursive machine* m*,*

$$\Sigma(p : \mathbb{N}), \Pi(y : \mathbb{N}), (\{p\}(y) = \mathrm{eval}(m, \langle p, y\rangle))\,.$$

Proof. Let $f : \mathbb{N} \to \mathbb{N}$ be defined by

$$f(x) \stackrel{\mathrm{def}}{=} s\langle e_m, x\rangle.$$

This function f is primitive recursive, so there is a total recursive machine m computing f. By Rogers' fixed point theorem, there is then a p such that $\{p\} = \{m(p)\}^*$. Then,

$$\{p\}(y) = \{m(p)\}^*(y) = \{f(p)\}(y) = \{s\langle e_m, p\rangle\}(y) = m\langle p, y\rangle. \qquad \blacktriangleleft$$

These two theorems have versions from the perspective of computability as property which follow as corollaries.

▶ **Corollary 14.** *For any total computable function* $f : \mathbb{N} \to \mathcal{L}\,\mathbb{N}$, $\exists(n : \mathbb{N}), \{n\} = \{f(n)\}^*$.

▶ **Corollary 15.** *For any computable function* $f : \mathbb{N} \to \mathcal{L}\,\mathbb{N}$,

$$\exists(p : \mathbb{N}), \Pi(y : \mathbb{N}), (\{p\}(y) = f(\langle p, y\rangle))\,.$$

A subset A of $\mathbb{N}$ is *recursive* if its characteristic function $\mathbb{N} \to \Omega$ factors through the embedding $2 \to \Omega$ via a computable function $\mathbb{N} \to 2$. It is *recursively enumerable* if it is the image of some computable function $f : \mathbb{N} \to \mathbb{N}$. As with primitive recursive predicates, any recursive set is complemented. We get the following form of a classical result.

▶ **Theorem 16.** *A subset* A *of* $\mathbb{N}$ *is recursive if and only if it is complemented and both* A *and its complement are recursively enumerable.*

Likewise we have the classical characterization of recursively enumerable sets.

▶ **Theorem 17.** *A subset of* $\mathbb{N}$ *is r.e. iff it is the domain of a computable function.*

A slightly better classical characterization requires some modification to be true constructively.

▶ **Theorem 18.** *If* A *is an inhabited subset of* $\mathbb{N}$*, then the following are equivalent.*

1. *A is the domain of a computable partial function.*
2. *A is the image of a computable partial function.*
3. *A is the image of a computable total function.*
4. *A is the image of a primitive recursive function.*

3.4 Not all partial functions are computable

As discussed in Section 2.8, it is consistent in topos logic that every Rosolini partial function is computable. However, the statement that every partial function $\mathbb{N} \to \mathcal{L}\,\mathbb{N}$ is computable is provably false in univalent type theory. In fact, if the partial function $f : \mathbb{N} \to \mathcal{L}\,\mathbb{N}$ defined by

$$\begin{aligned} \text{extent}\, f(x) &\stackrel{\text{def}}{=} (\{x\}(x) = \bot), \\ \text{value}(f(x))(p) &\stackrel{\text{def}}{=} 0 \end{aligned}$$

were computable, this would mean that the complement of the Halting Set is recursively enumerable.

3.5 Dominances and semidecidable propositions

We now compare Rosolini and semidecidable propositions, using the following technical lemma.

▶ **Lemma 19.** *If* X *is a type and* $P : X \to \mathcal{U}$ *is a family of types over* X *such that* $\text{isProp}\,(\Sigma(x : X), \|P(x)\|)$*, then* $(\Sigma(x : X), \|P(x)\|) = \|\Sigma(x : X), P(x)\|$.

Proof. By propositional extensionality, it is enough to construct functions in both directions. Right to left follows from the universal property of truncations. For the other, we have

$$((\Sigma(x : X), \|P(x)\|) \to \|\Sigma(x : X), P(x)\|) \simeq \Pi(x : X), (\|P(x)\| \to \|\Sigma(x : X), P(x)\|)\,.$$

and functoriality of truncation gives us an inhabitant of the right-hand type. ◀

Recall the relation R_m from Section 3.2 used in the definition of $\mathrm{eval}(m)$.

▶ **Theorem 20.** *For any machine* $m : \mathrm{ATM}$ *and any* $x, y : \mathbb{N}$*, the types* $R_m(x, y)$ *and* $\Sigma(y : \mathbb{N}), R_m(x, y)$ *are Rosolini propositions.*

Proof. For the first type, let $g : \mathbb{N} + \mathbb{N} \to 2$ be the function which takes value 1 on $\mathrm{inr}\, y$ and 0 otherwise. Define $\alpha_k \stackrel{\mathrm{def}}{=} g(\mathrm{run}_k(m, x))$. We have

$$(\alpha_k = 1) \simeq (g(\mathrm{run}_k(m, x)) = y),$$

so $R_m(x, y) \simeq \exists(k : \mathbb{N}), \alpha_k = 1$.

For the second type, let $h : \mathbb{N} + \mathbb{N} \to 2$ be the function which is 0 on $\mathrm{inl}\, j$ and 1 on $\mathrm{inr}\, j$ for all j, and again take $\alpha_k = h(\mathrm{run}_k(m, x))$. We have that

$$(\alpha_k = 1) \simeq (\Sigma(y : \mathbb{N}), \mathrm{run}_k(m, x) = \mathrm{inr}\, y).$$

Summing over all $k : \mathbb{N}$, rearranging and truncating takes us to

$$(\exists(k : \mathbb{N}), \alpha_k = 1) \simeq (\exists(y : \mathbb{N}), \Sigma(k : \mathbb{N}), \mathrm{run}_k(m, x) = \mathrm{inr}\, y).$$

By our technical lemma, we then have

$$(\exists(k : \mathbb{N}), \alpha_k = 1) \simeq \Sigma(y : \mathbb{N}), \exists(k : \mathbb{N}), \mathrm{run}_k(m, x) = \mathrm{inr}\, y. \qquad \blacktriangleleft$$

Moreover, the semidecidable propositions are exactly those which arise as the value of a program. That is,

▶ **Theorem 21.** *For all* $A : \mathcal{U}$

$$\mathrm{isSemiDecidable}(A) \iff \exists(f : \mathbb{N} \rightharpoonup \mathbb{N}), \mathrm{isComputable}(f) \times (A = \mathrm{extent}(f(0))).$$

Proof. Suppose we are given a function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ with a recursive machine t such that $A = \mathrm{extent}(f(0))$. We may define

$$\alpha(n) = \begin{cases} 0 & \text{if } \mathrm{run}_n(t, 0) = \mathrm{inr}\, k \text{ for some } k, \text{ and } \alpha(m) = 1 \text{ for } m < n, \\ 1 & \text{otherwise.} \end{cases}$$

It is easy to see that α is recursive, that $\langle \alpha \rangle$ is a proposition and that $A = \langle \alpha \rangle$.

Conversely, suppose we are given a total recursive α. Consider the constant function $f(x) = \mu k.(\alpha(k) = 0)$. Since $\alpha(k) = 0$ is recursive, this is recursive and it is clear that $\mathrm{extent}(f(0)) = A$. The result follows from the universal property of truncations. $\blacktriangleleft$

4 Related work, discussion, conjectures, questions and further work

Capretta [7] introduced a certain *delay monad*, and Chapman, Uustalu and Veltri [8] considered its quotient by weak bisimilarity. In order to show that the result is again a monad, they used countable choice. This appearance of choice should be no accident: we conjecture that the quotient constructed in [8] is equivalent to the Rosolini lifting, which, as we saw, also requires some amount of countable choice to be a monad. As we have seen, countable choice is also needed to show that the Rosolini lifting is closed under countable ascending joins.

As discussed in the introduction, we would like to find a notion of partial function which can be seen to be closed under composition (the dominance axiom (D5)), without choice,

before we attempt to consistently postulate that all such partial functions from $\mathbb{N}$ to $\mathbb{N}$ are computable.

Altenkirch, Danielsson and Kraus [21] consider a higher inductive-inductive definition of a partiality monad with a constructor that closes under countable ascending joins, without choice, and show that, in the presence of countable choice, their construction is equivalent to the above construction [8], and hence would also be equivalent to the Rosolini lifting if the above conjecture is true. It is not hard to see that this construction can be equivalently described as the lifting induced by the dominance obtained by closing the Rosolini propositions under Σ and countable ascending joins.

However, this doesn't work for our purposes. In fact, consider again the Kleene-bracket function $\{-\} : \mathbb{N} \to (\mathbb{N} \to \mathcal{L}\,\mathbb{N})$ that associates a partial function to a Gödel number of a Turing machine. As discussed above, the extents of definition of the partial functions in the image of the Kleene-bracket are precisely the semidecidable propositions, which form a *subset* of the Rosolini propositions. Thus, enlarging the set of Rosolini propositions doesn't help.

Also, it doesn't seem to be possible to show that the subset consisting of the semidecidable propositions form a dominance without choice as above. Nevertheless, without any form of choice, we have that the partial functions in the image of Kleene-bracket, namely the computable functions, *are* closed under composition. It seems that even for $d = \mathrm{isSemiDecidable}$, in the absence of enough choice, the function space $\mathbb{N} \to \mathcal{L}_d\,\mathbb{N}$ may in principle contain more than the computable partial functions $\mathbb{N} \to \mathcal{L}\,\mathbb{N}$, as in Section 3.4 above. This motivates the following considerations.

The function $D \stackrel{\mathrm{def}}{=} \mathrm{RosoliniStructure}$ satisfies the dominance axioms with the exception of (D2), where (D4) and (D5) then become structure rather than merely property. Crucially, a function (D5) that provides closure under Σ can be constructed without any form of choice. This induces a lift monad $\mathcal{L}_D$ with the same constructions discussed in Section 2, which turns out to be equivalent to Capretta's delay monad. With $d \stackrel{\mathrm{def}}{=} \mathrm{isRosolini}$, we have a map $\mathcal{L}_D(Y) \to \mathcal{L}_d(Y)$ that truncates structure, which in turn induces a function $(X \to \mathcal{L}_D(Y)) \to (X \to \mathcal{L}_d(Y))$ by post-composition. We refer to the maps in the image $D(X, Y)$ of this function as *disciplined*. Although the maps $X \to \mathcal{L}_d(Y)$ are not closed under Kleisli composition unless the Rosolini propositions form a dominance, we have that the disciplined ones are. Moreover, it is easy to see that the maps $\mathbb{N} \to \mathcal{L}_d(\mathbb{N})$ in the image of Kleene–Bracket are all disciplined. Perhaps it is consistent that the converse also holds, which amounts to saying that all disciplined maps $D(\mathbb{N}, \mathbb{N})$ are computable. In other words, we speculate that the statement "all disciplined partial functions $\mathbb{N} \rightharpoonup \mathbb{N}$ are Turing computable" is consistent with univalent type theory.

Acknowledgements. We thank Thierry Coquand and Andrew Swan for discussions about countable choice, Nicolai Kraus and Ian Orton for comments on a draft version, Mike Shulman for discussions about dominances, and Martin Hyland, John Longley and Pino Rosolini for discussions about dominances and the effective topos.

References

1 Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006.

2 Andrej Bauer and Davorin Lešnik. Metric spaces in synthetic topology. *Annals of Pure and Applied Logic*, 163(2):87–100, 2012. Third Workshop on Formal Topology. `doi:10.1016/j.apal.2011.06.017`.

3 Ana Bove. *General Recursion in Type Theory*, pages 39–58. Springer, 2003.

4 Ana Bove and Venanzio Capretta. *Nested General Recursion and Partiality in Type Theory*, pages 121–125. Springer, 2001.

5 Ana Bove and Venanzio Capretta. *A Type of Partial Recursive Functions*, pages 102–117. Springer, Berlin, Heidelberg, 2008.

6 Douglas Bridges and Fred Richman. *Varieties of constructive mathematics*, volume 97 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1987.

7 Venancio Capretta. General recursion via coinductive types. *Logical Methods in Computer Science*, 15:1–28, 2005.

8 James Chapman, Tarmo Uustalu, and Niccolò Veltri. *Quotienting the Delay Monad by Weak Bisimilarity*, pages 110–125. Springer, Cham, 2015.

9 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. arXiv, November 2016. URL: `https://arxiv.org/abs/1611.02108`.

10 Thierry Coquand, Bassel Mannaa, and Fabian Ruch. Stack semantics of type theory. arXiv, 2017. URL: `https://arxiv.org/abs/1701.02571`.

11 J. Martin E. Hyland. The effective topos. *Studies in Logic and the Foundations of Mathematics*, 110:165–216, 1982.

12 J. M. E. Hyland. *First steps in synthetic domain theory*, pages 131–156. Springer, 1991.

13 S. Kleene. Introduction to Metamathematics, 1952.

14 N. Kraus, M. H. Escardó, T. Coquand, and T. Altenkirch. Notions of anonymous existence in Martin-Löf type theory. *Logical Methods in Computer Science*, 2017.

15 Piergiorgio Odifreddi. *Classical Recursion Theory*. Elsevier, 1989.

16 B. Reus and Th. Streicher. *General synthetic domain theory – A logical approach (extended abstract)*, pages 293–313. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. `doi:10.1007/BFb0026995`.

17 Egbert Rijke. Homotopy type theory. Master's thesis, Utrecht University, 2012. URL: `"https://hottheory.files.wordpress.com/2012/08/hott2.pdf"`.

18 Egbert Rijke. The join contruction. arXiv:1701.07538, Jan 2017. URL: `https://arxiv.org/abs/1701.07538`.

19 H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987.

20 G. Rosolini. *Continuity and Effectiveness in Topoi*. PhD thesis, University of Oxford, 1986. URL: `ftp://ftp.disi.unige.it/person/RosoliniG/papers/coneit.ps.gz`.

21 Nils Anders Danielsson Thorsten Altenkirch and Nicolai Kraus. Partiality, revisited: The partiality monad as a quotient inductive-inductive type. In *FoSSaCS Proceedings*, 2017.

22 A. S. Troelstra. A note on non-extensional operations in connection with continuity and recursiveness. *Indag. Math.*, 39(5):455–462, 1977.

23 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

24 Jaap van Oosten and Alex K. Simpson. Axioms and (counter)examples in synthetic domain theory. *Annals of Pure and Applied Logic*, 104(1):233–278, 2000. `doi:10.1016/S0168-0072(00)00014-2`.

25 V. Voevodsky. Resizing rules. Invited talk at Types'2011, Bergen, September 2011. URL: `http://www.math.ias.edu/vladimir/files/2011_Bergen.pdf`.

Polishness of Some Topologies Related to Automata

Olivier Carton[1], Olivier Finkel[2], and Dominique Lecomte[3]

1 **Université Paris Diderot, LIAFA, UMR 7089, Case 7014, Paris, France**
Olivier.Carton@liafa.univ-paris-diderot.fr
2 **Equipe de Logique Mathématique, Institut de Mathématiques de Jussieu – Paris Rive Gauche, CNRS et Université Paris, France**
finkel@math.univ-paris-diderot.fr
3 **Projet Analyse Fonctionnelle, Institut de Mathématiques de Jussieu – Paris Rive Gauche, Université Paris, France; and**
Université de Picardie, I.U.T. de l'Oise, site de Creil, Amiens, France
dominique.lecomte@upmc.fr

Abstract

We prove that the Büchi topology, the automatic topology, the alphabetic topology and the strong alphabetic topology are Polish, and provide consequences of this.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic, F.4.3 Formal Languages

Keywords and phrases Automata and formal languages, logic in computer science, infinite words, Büchi automaton, regular ω-language, Cantor space, finer topologies, Büchi topology, automatic topology, alphabetic topology, strong alphabetic topology, Polish topology

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.22

1 Introduction

This paper is a contribution to the study of the interactions between descriptive set theory and theoretical computer science. These interactions have already been the subject of many studies, see for instance [18, 32, 21, 30, 27, 25, 29, 6, 28, 7, 11, 9, 5].

In particular, the theory of automata reading infinite words, which is closely related to infinite games, is now a rich theory which is used for the specification and verification of non-terminating systems, see [12, 25]. The space $\Sigma^{\mathbb{N}}$ of infinite words over a finite alphabet Σ being equipped with the usual Cantor topology, a natural way to study the complexity of ω-languages accepted by various kinds of automata is to study their topological complexity, and particularly to locate them with regard to the Borel and the projective hierarchies.

However, as noticed in [26] by Schwarz and Staiger and in [15] by Hoffmann and Staiger, it turned out that for several purposes some other topologies on a space $\Sigma^{\mathbb{N}}$ are useful, for instance for studying fragments of first-order logic over infinite words or for a topological characterisation of random infinite words (see also [14]). In particular, Schwarz and Staiger studied four topologies on the space $\Sigma^{\mathbb{N}}$ of infinite words over a finite alphabet Σ which are all related to automata, and refine the Cantor topology on $\Sigma^{\mathbb{N}}$: the Büchi topology, the automatic topology, the alphabetic topology, and the strong alphabetic topology.

Recall that a topological space is Polish iff it is separable, i.e. contains a countable dense subset, and its topology is induced by a complete metric. Classical descriptive set theory is about the topological complexity of definable subsets of Polish topological spaces, as well

as the study of hierarchies of complexity (see [16, 25] for basic notions). The analytic sets, which are the projections of Borel sets, are of particular importance. Similar hierarchies of complexity are studied in effective descriptive set theory, which is based on the theory of recursive functions (see [24] for basic notions). The effective analytic subsets of the Cantor space $(2^{\mathbb{N}}, \tau_C)$ are highly related to theoretical computer science, in the sense that they coincide with the sets recognized by some special kind of Turing machines (see [31]).

In [26], Schwarz and Staiger prove that the Büchi topology, which is generated by the regular ω-languages, is metrizable. It is separable, by definition, because there are only countably many regular ω-languages. It remains to see that it is completely metrizable to see that it is Polish. This is one of the main results proved in this paper.

We now give some more details about the topologies studied by Schwarz and Staiger in [26] that we investigate in this paper. Let Σ be a finite alphabet with at least two symbols. We will consider the topology τ_δ on $\Sigma^{\mathbb{N}}$ generated by the set $\mathbb{B}_\delta$ of sets accepted by an unambiguous Büchi Turing machine. Let Σ^* be the set of finite sequences of elements of Σ. The following topologies on $\Sigma^{\mathbb{N}}$, related to automata, are considered in [26].

- The *Büchi topology* τ_B, generated by the set $\mathbb{B}_B$ of ω-regular languages,
- The *automatic topology* τ_A, generated by the set $\mathbb{B}_A$ of τ_C-closed ω-regular languages (this topology is remarkable because all τ_C-closed ω-regular languages are accepted by deterministic Büchi automata),
- The *alphabetic topology* τ_α, generated by the set $\mathbb{B}_\alpha$ of sets of the form $B_{w,A} := \{w\sigma \mid \sigma \in A^{\mathbb{N}}\}$, where $w \in \Sigma^*$ and $A \subseteq \Sigma$ (this topology is useful for investigations in restricted first-order theories for infinite words),
- The *strong alphabetic topology* τ_s, generated by the set $\mathbb{B}_s$ of sets of the form

$$S_{w,A} := \{w\sigma \mid \sigma \in A^{\mathbb{N}} \ \wedge \ \forall a \in A \ \ \forall k \in \mathbb{N} \ \ \exists i \geq k \ \ \sigma(i) = a\},$$

  where $w \in \Sigma^*$ and $A \subseteq \Sigma$ (this topology is derived from τ_α, and considered in [4], together with τ_α).

In [26], Schwarz and Staiger prove that these topologies are metrizable. We improve this result:

▶ **Theorem 1.** *Let $z \in \{C, \delta, B, A, \alpha, s\}$. Then τ_z is Polish.*

From this result, it is already possible to infer many properties of the space $\Sigma^{\mathbb{N}}$, where Σ is a finite alphabet, equipped with the Büchi topology. In particular, we first get some results about the σ-algebra generated by the ω-regular languages. It is stratified in a hierarchy of length ω_1 (the first uncountable ordinal) and there are universal sets at each level of this hierarchy. Notice that this σ-algebra coincides with the σ-algebra of Borel sets for the Cantor topology, and that a set is Borel for the Cantor topology if and only if it is Borel for the Büchi topology, but the levels of the Borel hierarchy differ for the two topologies. For instance an ω-regular set which is non-$\mathbf{\Pi}^0_2$ for the Cantor topology is clopen (i.e., $\mathbf{\Delta}^0_1$) for the Büchi topology. Therefore the results about the existence of universal sets at each level of the σ-algebra generated by the ω-regular languages are really new and interesting. We derive many other properties from the polishness of the Büchi topology.

2 Background

We first recall the notions required to understand fully the introduction and the sequel (see for example [25, 30, 16, 24]).

2.1 Theoretical computer science

A *Büchi automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_i, Q_f)$, where Σ is the input alphabet, Q is the finite state set, δ is the transition relation, Q_i and Q_f are the sets of initial and final states. The transition relation δ is a subset of $Q \times \Sigma \times Q$.

A *run* on some sequence $\sigma \in \Sigma^{\mathbb{N}}$ is a sequence $(q_n)_{i\in\mathbb{N}} \in Q^{\mathbb{N}}$ of states such that q_0 is initial ($q_0 \in Q_i$) and $(q_i, \sigma(i), q_{i+1})$ is a transition in δ for each $i \geq 0$. It is accepting if it visits infinitely often final states, that is, $q_i \in Q_f$ for infinitely many i. An input sequence σ is accepted if there exists an accepting run on α. The set of accepted inputs is denoted $L(\mathcal{A})$. A set of infinite words is called *ω-regular* if it is equal to $L(\mathcal{A})$ for some automaton $\mathcal{A}$.

A Büchi automaton is actually similar to a classical finite automaton. A finite word w of length n is accepted by an automaton $\mathcal{A}$ if there is sequence $(q_i)_{i\leq n}$ of $n+1$ states such that q_0 is initial ($q_0 \in Q_i$), q_n is final ($q_n \in Q_f$) and $(q_i, \sigma(i), q_{i+1})$ is a transition in δ for each $0 \leq i < n$. The set of accepted finite words is denoted by $U(\mathcal{A})$. A set of finite words is called *regular* if it is equal to $U(\mathcal{A})$ for some automaton $\mathcal{A}$.

Let us recall that the ω-power of a set U of finite words is defined by

$$U^\omega = \{\sigma \in \Sigma^{\mathbb{N}} \mid \exists\, (w_i)_{i\in\mathbb{N}} \in U^{\mathbb{N}} \text{ s.t. } \sigma = w_0 w_1 w_2 \cdots\}.$$

The ω-powers play a crucial role in the characterization of ω-regular languages (see [1]):

▶ **Theorem 2** (Büchi). *Let Σ be a finite alphabet, and $L \subseteq \Sigma^{\mathbb{N}}$. The following statements are equivalent:*

1. *L is ω-regular,*
2. *there are $2n$ regular languages $(U_i)_{i<n}$ and $(V_i)_{i<n}$ such that $L = \bigcup_{i<n} U_i V_i^\omega$.*

The closure properties of the class of ω-regular languages mentioned in the introduction are the following (see [25] and [26]). If Σ is a set and $w \in \Sigma^*$, then w defines the usual basic clopen set $N_w := \{\sigma \in \Sigma^{\mathbb{N}} \mid w \text{ is a prefix of} \sigma\}$ of the Cantor topology τ_C (so $\mathbb{B}_C := \{\emptyset\} \cup \{N_w \mid w \in \Sigma^*\}$ is a basis for τ_C).

▶ **Theorem 3** (Büchi). *The class of ω-regular languages contains the usual basic clopen sets and is closed under finite unions and intersections, taking complements, and projections.*

We now turn to the study of Turing machines (see [3, 30]).

A *Büchi Turing machine* is a tuple $\mathcal{M} = (\Sigma, Q, \Gamma, \delta, q_0, Q_f)$, where Σ and Γ are the input and tape alphabets satisfying $\Sigma \subseteq \Gamma$, Q is the finite state set, δ is the transition relation, q_0 is the initial state and Q_f is the set of final states. The relation δ is a subset of $(Q \times \Gamma) \times (Q \times \Gamma \times \{-1, 0, 1\})$.

A *configuration* of $\mathcal{M}$ is a triple (q, γ, j) where $q \in Q$ is the current state, $\gamma \in \Gamma^{\mathbb{N}}$ is the content of the tape and the non-negative integer $j \in \mathbb{N}$ is the position of the head on the tape.

Two configurations (q, γ, j) and (q', γ', j') of $\mathcal{M}$ are *consecutive* if there exists a transition $(q, a, q', b, d) \in \delta$ such that the following conditions are met:

1. $\gamma(j) = a$, $\gamma'(j) = b$ and $\gamma(i) = \gamma'(i)$ for each $i \neq j$. This means that the symbol a is the replaced by symbol b at position j and that all other symbols on the tape remain unchanged.
2. the two positions j and j' satisfy the equality $j' = j + d$.

A *run* of the machine $\mathcal{M}$ on some input $\sigma \in \Sigma^{\mathbb{N}}$ is a sequence $(p_i, \gamma_i, j_i)_{i\in\mathbb{N}}$ of consecutive configurations such that $p_0 = q_0$, $\gamma_0 = \sigma$ and $j_0 = 0$. It is *complete* if the head visits all positions of the tape. This means that for each integer N, the exists an integer i such that

$j_i \geq N$. The run is *accepting* if it visits infinitely often final states, that is, $p_i \in Q_f$ for infinitely many integers i. The ω-language accepted by $\mathcal{M}$ is the set of inputs σ such that there exists an accepting and complete run on σ.

Notice that other accepting conditions have been considered for the acceptance of infinite words by Turing machines, like the 1' or Muller ones (the latter one was firstly called 3-acceptance), see [3, 30]. Moreover several types of required behaviour on the input tape have been considered in the literature, see [31, 10, 8].

A *Büchi automaton* $\mathcal{A}$ is in fact a Büchi Turing machine whose head only move forwards. This means that each of its transition has the form (p, a, q, b, d) where $d = 1$. Note that the written symbol b does not matter since each position of the tape is just visited once and the symbol b is never read.

2.2 Descriptive set theory

Classical descriptive set theory takes place in Polish topological spaces.
We first recall that if d is a distance on a set X, and $(x_n)_{n\in\mathbb{N}}$ is a sequence of elements of X, then the sequence $(x_n)_{n\in\mathbb{N}}$ is called a Cauchy sequence if

$$\forall k \in \mathbb{N} \;\; \exists N \in \mathbb{N} \;\; \forall p, p' \geq N \;\; d(x_p, x_{p'}) < \frac{1}{2^k}.$$

In a topological space X whose topology is induced by a distance d, the distance d and the metric space (X, d) are said to be *complete* if every Cauchy sequence in X is convergent.

▶ **Definition 4.** A topological space X is a *Polish space* if it is

1. separable (there is a countable dense sequence (x_n) in X),
2. completely metrizable (there is a complete distance d on X which is compatible with the topology of X).

Effective descriptive set theory is based on the notion of recursive function. A function from $\mathbb{N}^k$ to $\mathbb{N}^l$ is said to be *recursive* if it is total and computable. By extension, a relation is called *recursive* if its characteristic function is recursive.

▶ **Definition 5.** A *recursive presentation* of a Polish space X is a pair $\big((x_n)_{n\in\mathbb{N}}, d\big)$ such that

1. $(x_n)_{n\in\mathbb{N}}$ is dense in X,
2. d is a compatible complete distance on X such that the following relations P and Q are recursive:

$$P(i,j,m,k) \iff d(x_i, x_j) \leq \frac{m}{k+1},$$
$$Q(i,j,m,k) \iff d(x_i, x_j) < \frac{m}{k+1}.$$

A Polish space X is *recursively presented* if there is a recursive presentation of it.

Note that the formula $(p, q) \mapsto 2^p(2q + 1) - 1$ defines a recursive bijection $\mathbb{N}^2 \to \mathbb{N}$. One can check that the coordinates of the inverse map are also recursive. They will be denoted $n \mapsto (n)_0$ and $n \mapsto (n)_1$ in the sequel. These maps will help us to define some of the basic effective classes.

▶ **Definition 6.** Let $\big((x_n)_{n\in\mathbb{N}}, d\big)$ be a recursive presentation of a Polish space X.

1. We fix a countable basis of X: $B(X, n)$ is the open ball $B_d(x_{(n)_0}, \frac{(n)_0}{((n)_1)_1+1})$.

2. A subset S of X is *semirecursive*, or *effectively open* (denoted $S \in \Sigma^0_1$) if

$$S = \bigcup_{n\in\mathbb{N}} B(X, f(n)),$$

for some recursive function f.
3. A subset S of X is *effectively closed* (denoted $S \in \Pi^0_1$) if its complement $\neg S$ is semirecursive.
4. One can check that a product of two recursively presented Polish spaces has a recursive presentation, and that the Baire space $\mathbb{N}^{\mathbb{N}}$ has a recursive presentation. A subset S of X is *effectively analytic* (denoted $S \in \Sigma^1_1$) if there is a Π^0_1 subset C of $X \times \mathbb{N}^{\mathbb{N}}$ such that

$$S = \pi_0[C] := \{x \in X \mid \exists \alpha \in \mathbb{N}^{\mathbb{N}} \ \ (x, \alpha) \in C\}.$$

5. A subset S of X is *effectively co-analytic* (denoted $S \in \Pi^1_1$) if its complement $\neg S$ is effectively analytic, and *effectively Borel* if it is in Σ^1_1 and Π^1_1 (denoted $S \in \Delta^1_1$).
6. We will also use the following *relativized classes*: if X, Y are recursively presented Polish spaces and $y \in Y$, then we say that $A \subseteq X$ is in $\Sigma^1_1(y)$ if there is $S \in \Sigma^1_1(Y \times X)$ such that $A = S_y := \{x \in X \mid (y, x) \in S\}$. The class $\Pi^1_1(y)$ is defined similarly. We also set $\Delta^1_1(y) := \Sigma^1_1(y) \cap \Pi^1_1(y)$.

The crucial link between the effective classes and the classical corresponding classes is as follows: the class of analytic (resp., co-analytic, Borel) subsets of Y is equal to $\bigcup_{\alpha\in\mathbb{N}^{\mathbb{N}}} \Sigma^1_1(\alpha)$ (resp., $\bigcup_{\alpha\in\mathbb{N}^{\mathbb{N}}} \Pi^1_1(\alpha)$, $\bigcup_{\alpha\in\mathbb{N}^{\mathbb{N}}} \Delta^1_1(\alpha)$). This allows to use effective descriptive set theory to prove results of classical type. In the sequel, when we consider an effective class in some $\Sigma^{\mathbb{N}}$ with Σ finite, we will always use a fixed recursive presentation associated with the Cantor topology. The following result is proved in [31], see also [8]:

▶ **Theorem 7.** *Let Σ be a finite alphabet, and $L \subseteq \Sigma^{\mathbb{N}}$. The following statements are equivalent:*
1. *$L = L(\mathcal{M})$ for some Büchi Turing machine $\mathcal{M}$,*
2. *$L \in \Sigma^1_1$.*

We now recall the *Choquet game* played by two players on a topological space X. Players 1 and 2 play alternatively. At each turn i, Player 1 plays by choosing an open subset U_i and a point $x_i \in U_i$ such that $U_i \subseteq V_{i-1}$, where V_{i-1} has been chosen by Player 2 at the previous turn. Player 2, plays by choosing an open subset V_i such that $x_i \in V_i$ and $V_i \subseteq U_i$. Player 2 wins the game if $\bigcap_{i\in\mathbb{N}} V_i \neq \emptyset$. We now recall some classical notions of topology.

▶ **Definition 8.** A topological space X is said to be
- T_1 if every singleton of X is closed,
- *regular* if for every point of X and every open neighborhood U of x, there is an open neighborhood V of x with $\overline{V} \subseteq U$,
- *second countable* if its topology has a countable basis,
- *zero-dimensional* if there is a basis made of clopen sets,
- *strong Choquet* if X is not empty and Player 2 has a winning strategy in the Choquet game.

Note that every zero-dimensional space is regular. The following result is 8.18 in [16].

▶ **Theorem 9** (Choquet)**.** *A nonempty, second countable topological space is Polish if and only if it is T_1, regular, and strong Choquet.*

Let X be a nonempty recursively presented Polish space. The *Gandy-Harrington topology* on X is generated by the Σ^1_1 subsets of X, and denoted Σ_X. By Theorem 7, this topology is also related to automata and Turing machines. As there are some effectively analytic sets whose complement is not analytic, the Gandy-Harrington topology is not metrizable (in fact not regular) in general (see [3E.9] in [24]). In particular, it is not Polish.

3 Proof of the main result

The proof of Theorem 1 is organized as follows. We provide below four properties which ensure that a given topological space is strong Choquet. Then we use Theorem 9 to prove that the considered spaces are indeed Polish.

Let Σ be a countable alphabet. The set $\Sigma^{\mathbb{N}}$ is equipped with the product topology of the discrete topology on Σ, unless another topology is specified. This topology is induced by a natural metric, called the *prefix metric* which is defined as follows. For $\sigma \neq \sigma' \in \Sigma^{\mathbb{N}}$, the distance d is given by

$$d(\sigma, \sigma') = 2^{-r} \quad \text{where} \quad r = \min\{n \mid \sigma(n) \neq \sigma'(n)\}.$$

When Σ is finite this topology is the classical Cantor topology. When Σ is countably infinite the topological space is homeomorphic to the Baire space $\mathbb{N}^{\mathbb{N}}$.

Let Σ and Γ be two alphabets. The function which maps each pair $(\sigma, \gamma) \in \Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ to the element $(\sigma(0), \gamma(0)), (\sigma(1), \gamma(1)), \dots$ of $(\Sigma \times \Gamma)^{\mathbb{N}}$ is a homeomorphism between $\Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ and $(\Sigma \times \Gamma)^{\mathbb{N}}$ allowing us to identify these two spaces.

If Σ is a set, $\sigma \in \Sigma^{\mathbb{N}}$ and $l \in \mathbb{N}$, then $\sigma|l$ is the prefix of σ of length l.

We set $2 := \{0, 1\}$ and $\mathbb{P}_\infty := \{\alpha \in 2^{\mathbb{N}} \mid \forall k \in \mathbb{N} \ \ \exists i \geq k \ \ \alpha(i) = 1\}$. This latter set is simply the set of infinite words over the alphabet $2 := \{0, 1\}$ having infinitely many symbols 1.

We will work in the spaces of the form $\Sigma^{\mathbb{N}}$, where Σ is a finite set with at least two elements. We will fix a topology τ_Σ on $\Sigma^{\mathbb{N}}$, and a basis $\mathbb{B}_\Sigma$ for τ_Σ. We consider the following properties of the family $(\tau_\Sigma, \mathbb{B}_\Sigma)_\Sigma$, using the previous identification:

(P1) $\mathbb{B}_\Sigma$ contains the usual basic clopen sets N_w,

(P2) $\mathbb{B}_\Sigma$ is closed under finite unions and intersections,

(P3) $\mathbb{B}_\Sigma$ is closed under projections, in the sense that if Γ is a finite set with at least two elements and $L \in \mathbb{B}_{\Sigma \times \Gamma}$, then $\pi_0[L] \in \mathbb{B}_\Sigma$,

(P4) for each $L \in \mathbb{B}_\Sigma$ there is a closed subset C of $\Sigma^{\mathbb{N}} \times \mathbb{P}_\infty$ (i.e. C is the intersection of a closed subset of the Cantor space $\Sigma^{\mathbb{N}} \times 2^{\mathbb{N}}$ with $\Sigma^{\mathbb{N}} \times \mathbb{P}_\infty$) which is in $\mathbb{B}_{\Sigma \times 2}$, and such that $L = \pi_0[C]$.

▶ **Theorem 10.** *Assume that the family $(\tau_\Sigma, \mathbb{B}_\Sigma)_\Sigma$ satisfies the properties (P1)-(P4). Then the topologies τ_Σ are strong Choquet.*

Proof. We first describe a strategy τ for Player 2. Player 1 first plays $\sigma_0 \in \Sigma^{\mathbb{N}}$ and a τ_Σ-open neighborhood U_0 of σ_0. Let L_0 in $\mathbb{B}_\Sigma$ with $\sigma_0 \in L_0 \subseteq U_0$. Property (P4) gives C_0 with $L_0 = \pi_0[C_0]$. This gives $\alpha_0 \in \mathbb{P}_\infty$ such that $(\sigma_0, \alpha_0) \in C_0$. We choose $l^0_0 \in \mathbb{N}$ big enough to ensure that if $s^0_0 := \alpha_0|l^0_0$, then s^0_0 has at least a coordinate equal to 1. We set $w_0 := \sigma_0|1$ and $V_0 := \pi_0[C_0 \cap (N_{w_0} \times N_{s^0_0})]$. By properties (P1)-(P3), V_0 is in $\mathbb{B}_\Sigma$ and thus τ_Σ-open. Moreover, $\sigma_0 \in V_0 \subseteq L_0 \subseteq U_0$, so that Player 2 respects the rules of the game if he plays V_0.

Now Player 1 plays $\sigma_1 \in V_0$ and a τ_Σ-open neighborhood U_1 of σ_1 contained in V_0. Let L_1 in $\mathbb{B}_\Sigma$ with $\sigma_1 \in L_1 \subseteq U_1$. Property (P4) gives C_1 with $L_1 = \pi_0[C_1]$. This gives $\alpha_1 \in \mathbb{P}_\infty$ such that $(\sigma_1, \alpha_1) \in C_1$. We choose $l^1_0 \in \mathbb{N}$ big enough to ensure that if $s^1_0 := \alpha_1|l^1_0$, then s^1_0 has at least one coordinate equal to 1. As $\sigma_1 \in V_0$, there is $\alpha'_0 \in \mathbb{P}_\infty$

such that $(\sigma_1, \alpha'_0) \in C_0 \cap (N_{w_0} \times N_{s^0_0})$. We choose $l^0_1 > l^0_0$ big enough to ensure that if $s^0_1 := \alpha'_0|l^0_1$, then s^0_1 has at least two coordinates equal to 1. We set $w_1 := \sigma_1|2$ and $V_1 := \pi_0[C_0 \cap (N_{w_1} \times N_{s^0_1})] \cap \pi_0[C_1 \cap (N_{w_0} \times N_{s^1_0})]$. Here again, V_1 is τ_Σ-open. Moreover, $\sigma_1 \in V_1 \subseteq U_1$ and Player 2 can play V_1.

Next, Player 1 plays $\sigma_2 \in V_1$ and a τ_Σ-open neighborhood U_2 of σ_2 contained in V_1. Let L_2 in $\mathbb{B}_\Sigma$ with $\sigma_2 \in L_2 \subseteq U_2$. Property (P4) gives C_2 with $L_2 = \pi_0[C_2]$. This gives $\alpha_2 \in \mathbb{P}_\infty$ such that $(\sigma_2, \alpha_2) \in C_2$. We choose $l^2_0 \in \mathbb{N}$ big enough to ensure that if $s^2_0 := \alpha_2|l^2_0$, then s^2_0 has at least one coordinate equal to 1. As $\sigma_2 \in V_1$, there is $\alpha'_1 \in \mathbb{P}_\infty$ such that $(\sigma_2, \alpha'_1) \in C_1 \cap (N_{w_0} \times N_{s^1_0})$. We choose $l^1_1 > l^1_0$ big enough to ensure that if $s^1_1 := \alpha'_1|l^1_1$, then s^1_1 has at least two coordinates equal to 1. As $\sigma_2 \in V_1$, there is $\alpha''_0 \in \mathbb{P}_\infty$ such that $(\sigma_2, \alpha''_0) \in C_0 \cap (N_{w_1} \times N_{s^0_1})$. We choose $l^0_2 > l^0_1$ big enough to ensure that if $s^0_2 := \alpha''_0|l^0_2$, then s^0_2 has at least three coordinates equal to 1. We set $w_2 := \sigma_2|3$ and $V_2 := \pi_0[C_0 \cap (N_{w_2} \times N_{s^0_2})] \cap \pi_0[C_1 \cap (N_{w_1} \times N_{s^1_1})] \cap \pi_0[C_2 \cap (N_{w_0} \times N_{s^2_0})]$. Here again, V_2 is τ_Σ-open. Moreover, $\sigma_2 \in V_2 \subseteq U_2$ and Player 2 can play V_2.

If we go on like this, we build $w_l \in \Sigma^{l+1}$ and $s^n_l \in 2^*$ such that $w_0 \subseteq w_1 \subseteq \ldots$ and $s^n_0 \subsetneqq s^n_1 \subsetneqq \ldots$ This allows us to define $\sigma := \lim_{l\to\infty} w_l \in \Sigma^{\mathbb{N}}$ and, for each $n \in \mathbb{N}$, $\alpha_n := \lim_{l\to\infty} s^n_l \in 2^{\mathbb{N}}$. Note that $\alpha_n \in \mathbb{P}_\infty$ since s^n_l has at least $l+1$ coordinates equal to 1. As (σ, α_n) is the limit of (w_l, s^n_l) as l goes to infinity and $N_{w_l} \times N_{s^n_l}$ meets C_n (which is closed in $\Sigma^{\mathbb{N}} \times \mathbb{P}_\infty$), $(\sigma, \alpha_n) \in C_n$. Thus

$$\sigma \in \bigcap_{n\in\mathbb{N}} \pi_0[C_n] = \bigcap_{n\in\mathbb{N}} L_n \subseteq \bigcap_{n\in\mathbb{N}} U_n \subseteq \bigcap_{n\in\mathbb{N}} V_n,$$

so that τ is winning for Player 2. ◀

3.1 The Gandy-Harrington topology

We have already mentioned that the Gandy-Harrington topology is not Polish in general. However, it is almost Polish as it fulfills the four properties (P1)–(P4).

Let Σ be a finite alphabet with at least two elements and let X be the space $\Sigma^{\mathbb{N}}$ equipped with the topology $\tau_\Sigma := \Sigma_X$ generated by the family $\mathbb{B}_\Sigma$ of Σ^1_1 subsets of X. Note that the assumption of Theorem 10 are satisfied. Indeed, (P1)-(P3) come from 3E.2 in [24]. For (P4), let F be a Π^0_1 subset of $X \times \mathbb{N}^{\mathbb{N}}$ such that $L = \pi_0[F]$. Let φ be the function from $\mathbb{N}^{\mathbb{N}}$ to $2^{\mathbb{N}}$ defined by $\varphi(\beta) = 0^{\beta(0)}10^{\beta(1)}1\ldots$. Note that φ is a homeomorphism from $\mathbb{N}^{\mathbb{N}}$ onto $\mathbb{P}_\infty$, and recursive (which means that the relation $\varphi(\beta) \in N(2^{\mathbb{N}}, n)$ is semirecursive in β and n). This implies that $C := (\mathrm{Id} \times \varphi)[F]$ is suitable (see 3E.2 in [24]).

Note that τ_Σ is second countable since there are only countably many Σ^1_1 subsets of X (see 3F.6 in [24]), T_1 since it is finer than the usual topology by the property (P1), and strong Choquet by Theorem 10.

One can show that there is a dense basic open subset Ω_X of (X, τ_Σ) such that $S \cap \Omega_X$ is a clopen subset of (Ω_X, τ_Σ) for each Σ^1_1 subset S of X (see [19]). In particular, (Ω_X, τ_Σ) is zero-dimensional, and regular. As it is, just like (X, τ_Σ), second countable, T_1 and and strong Choquet, (Ω_X, τ_Σ) is a Polish space, by Theorem 9.

3.2 The Büchi topology

Let Σ be a finite alphabet with at least two symbols, and X be the space $\Sigma^{\mathbb{N}}$ equipped with the Büchi topology τ_B generated by the family $\mathbb{B}_B$ of ω-regular languages in X. Theorem 29 in [26] shows that τ_B is metrizable. We now give a distance which is compatible with τ_B. This metric was used in [14] (Theorem 2 and Lemma 21 and several corollaries following

Lemma 21). A similar argument for subword metrics is in [[15], Section 4]. If $\mathcal{A}$ is a Büchi automaton, then we denote $|\mathcal{A}|$ the number of states of $\mathcal{A}$. We say that a Büchi automaton **separates** x and y iff

$$\big(x \in L(\mathcal{A}) \wedge y \notin L(\mathcal{A})\big) \vee \big(y \in L(\mathcal{A}) \wedge x \notin L(\mathcal{A})\big).$$

The distance δ on $\Sigma^{\mathbb{N}}$ is then defined as follows: for $x, y \in \Sigma^{\mathbb{N}}$, $\delta(x,y) = \begin{cases} 0 \text{ if } x = y, \\ 2^{-n} \text{ if } x \neq y, \end{cases}$ where $n := \min\{|\mathcal{A}| \mid \mathcal{A} \text{ is a Büchi automaton which separates } x \text{ and } y\}$. We now describe some properties of the map δ. This is the occasion to illustrate the notion of complete metric.

▶ **Proposition 11.**
1. *the map δ defines a distance on $\Sigma^{\mathbb{N}}$,*
2. *the distance δ is compatible with τ_B,*
3. *the distance δ is not complete.*

Proof. 1. If $x, y \in \Sigma^{\mathbb{N}}$, then $\delta(x,y) = \delta(y,x)$, by definition of δ. Let $x, y, z \in \Sigma^{\mathbb{N}}$, and assume that $\delta(x,y) + \delta(y,z) < \delta(x,z) = 2^{-n}$. Then $\delta(x,y) < 2^{-n}$ and $\delta(y,z) < 2^{-n}$ hold. In particular, if $\mathcal{A}$ is a Büchi automaton with n states then it does not separate x and y and similarly it does not separate y and z. Thus either $x, y, z \in L(\mathcal{A})$ or $x, y, z \notin L(\mathcal{A})$. This implies that the Büchi automaton $\mathcal{A}$ does not separate x and z. But this holds for every Büchi automaton with n states and then $\delta(x,z) < 2^{-n}$. This leads to a contradiction and thus $\delta(x,z) \leq \delta(x,y) + \delta(y,z)$ for all $x, y, z \in \Sigma^{\mathbb{N}}$. This shows that δ is a distance on $\Sigma^{\mathbb{N}}$.

2. Recall that an open set for this topology is a union of ω-languages accepted by Büchi automata. Let then $L(\mathcal{A})$ be an ω-language accepted by a Büchi automaton $\mathcal{A}$ having n states, and $x \in L(\mathcal{A})$. We now show that the open ball $B(x, 2^{-(n+1)})$ with center x and δ-radius $2^{-(n+1)}$ is a subset of $L(\mathcal{A})$. Indeed, if $\delta(x,y) < 2^{-(n+1)} < 2^{-n}$, then x and y cannot be separated by any Büchi automaton with n states, and thus $y \in L(\mathcal{A})$. This shows that $L(\mathcal{A})$ (and therefore any open set for τ_B) is open for the topology induced by the distance δ. Conversely, let $B(x,r)$ be an open ball for the distance δ, where $r > 0$ is a positive real. It is clear from the definition of the distance δ that we may only consider the case $r = 2^{-n}$ for some natural number n. Then $y \in B(x, 2^{-n})$ if and only if x and y cannot be separated by any Büchi automaton with $p \leq n$ states. Therefore the open ball $B(x, 2^{-n})$ is the intersection of the regular ω-languages $L(\mathcal{A}_i)$ for Büchi automata $\mathcal{A}_i$ having $p \leq n$ states and such that $x \in L(\mathcal{A}_i)$ and of the regular ω-languages $\Sigma^{\mathbb{N}} \setminus L(\mathcal{B}_i)$ for Büchi automata $\mathcal{B}_i$ having $p \leq n$ states and such that $x \notin L(\mathcal{B}_i)$. The class of regular ω-languages being closed under complementation and finite intersection, the open ball $B(x, 2^{-n})$ is actually a regular ω-language and thus an open set for τ_B.

3. Without loss of generality, we set $\Sigma = \{0,1\}$ and we consider, for a natural number $n \geq 1$, the ω-word $X_n = 0^{n!} \cdot 1 \cdot 0^{\omega}$ over the alphabet $\{0,1\}$ having only one symbol 1 after $n!$ symbols 0, where $n! := n \times (n-1) \times \cdots \times 2 \times 1$. Let now $m > n > k$ and $\mathcal{A}$ be a Büchi automaton with k states. Using a classical pumping argument, we can see that the automaton $\mathcal{A}$ cannot separate X_n and X_m. Indeed, assume first that $X_n \in L(\mathcal{A})$. Then, when reading the first k symbols 0 of X_n, the automaton enters at least twice in a same state q. This implies that there is a sequence of symbols 0 of length $p \leq k$ which can be added several times to the word X_n so that the resulting word will still be accepted by $\mathcal{A}$. Formally, any word $0^{n!+lp} \cdot 1 \cdot 0^{\omega}$, for a natural number $l \geq 1$, will be accepted by $\mathcal{A}$. In particular $m! = n! \times (n+1) \times \cdots \times m = n! + n! \times \Big(\big((n+1) \times \cdots \times m\big) - 1\Big)$ is of this form and thus $X_m \in L(\mathcal{A})$. A very similar pumping argument shows that if $X_m \in L(\mathcal{A})$, then $X_n \in L(\mathcal{A})$. This shows that $\delta(X_n, X_m) < 2^{-k}$ and finally that the sequence (X_n) is a Cauchy sequence

for the distance δ. On the other hand if this sequence was converging to an ω-word x then x should be the word 0^ω because τ_B is finer than τ_C. But 0^ω is an ultimately periodic word and thus it is an isolated point for τ_B. This would lead to a contradiction, and thus the distance δ is not complete because the sequence (X_n) is a Cauchy sequence which is not convergent. ◀

Proposition 11 gives a motivation for deriving Theorem 1 from Theorem 10. Note that the assumption of Theorem 10 are satisfied. Indeed, (P1)-(P3) come from Theorem 3. We now check (P4).

▶ **Lemma 12.** *Let Σ be a finite set with at least two elements, and $L \subseteq \Sigma^{\mathbb{N}}$ be an ω-regular language. Then there is a closed subset C of $\Sigma^{\mathbb{N}} \times \mathbb{P}_\infty$, which is ω-regular as a subset of $(\Sigma \times 2)^{\mathbb{N}}$ identified with $\Sigma^{\mathbb{N}} \times 2^{\mathbb{N}}$, and such that $L = \pi_0[C]$.*

Proof. Let $\mathcal{A} = (\Sigma, Q, \delta, Q_i, Q_f)$ be a Büchi automaton and let $L = L(\mathcal{A})$ be its set of accepted words. Let χ_f be the characteristic function of the final states. It maps each state q to 1 if $q \in Q_f$ and to 0 otherwise. The function χ_f is extended to $Q^{\mathbb{N}}$ by setting $\alpha = \chi_f((q_n)_{n\in\mathbb{N}})$ where $\alpha(n) = \chi_f(q_n)$. Note that a run ρ of $\mathcal{A}$ is accepting if and only if $\chi_f(\rho) \in \mathbb{P}_\infty$.

Let C be the subset of $\Sigma^{\mathbb{N}} \times \mathbb{P}_\infty$ defined by

$$C := \big\{(\sigma, \alpha) \in \Sigma^{\mathbb{N}} \times \mathbb{P}_\infty \mid \exists \rho \text{ run of } \mathcal{A} \text{ on } \sigma \text{ s. t. } \alpha = \chi_f(\rho)\big\}.$$

By definition of C, $L = \pi_0[C]$. Let K be the subset of $\Sigma^{\mathbb{N}} \times 2^{\mathbb{N}} \times Q^{\mathbb{N}}$ be defined by

$$K := \big\{(\sigma, \alpha, \rho) \in \Sigma^{\mathbb{N}} \times 2^{\mathbb{N}} \times Q^{\mathbb{N}} \mid \rho \text{ is a run of } \mathcal{A} \text{ on } \sigma \text{ s. t. } \alpha = \chi_f(\rho)\big\}.$$

Since K is compact as a closed subset of a compact space and $C = \pi_{\Sigma^{\mathbb{N}}\times 2^{\mathbb{N}}}[K] \cap (\Sigma^{\mathbb{N}} \times \mathbb{P}_\infty)$, the subset C is a closed subset of $\Sigma^{\mathbb{N}} \times \mathbb{P}_\infty$. It remains to show that C is indeed ω-regular. Let Δ be defined by

$$\Delta := \big\{\big(p, (a, \varepsilon), q\big) \in Q \times (\Sigma \times 2) \times Q \mid (p, a, q) \in \delta \wedge (\varepsilon = 1 \iff p \in Q_f)\big\}.$$

This allows us to define a Büchi automaton by $\mathcal{A}' := (\Sigma \times 2, Q, \Delta, Q_i, Q_f)$. Note that

$$\begin{aligned}(\sigma, \alpha) \in L(\mathcal{A}') &\Leftrightarrow \exists (s_i)_{i\in\mathbb{N}} \in Q^{\mathbb{N}} \ \big(s_0 \in Q_i \ \wedge\ \forall i \in \mathbb{N} \ \ (s_i, \big(\sigma(i), \alpha(i)\big), s_{i+1}) \in \Delta\big) \wedge \\ &\qquad\qquad \forall k \in \mathbb{N} \ \ \exists i \geq k \ \ s_i \in Q_f \\ &\Leftrightarrow \alpha \in \mathbb{P}_\infty \ \wedge\ \exists (s_i)_{i\in\mathbb{N}} \in Q^{\mathbb{N}} \ \big(s_0 \in Q_i \ \wedge\ \forall i \in \mathbb{N} \ \ (s_i, \sigma(i), s_{i+1}) \in \delta \wedge \\ &\qquad\qquad (\alpha(i) = 1 \Leftrightarrow s_i \in Q_f)\big) \\ &\Leftrightarrow (\sigma, \alpha) \in C.\end{aligned}$$

Thus $C = L(\mathcal{A}')$ is ω-regular. ◀

▶ **Corollary 13.** *Let Σ be a finite set with at least two elements. Then the Büchi topology τ_B is zero-dimensional and Polish.*

Proof. As there are only countably many possible automata (up to identifications), $\mathbb{B}_B$ is countable. This shows that τ_B is second countable. It is T_1 since it is finer than the usual topology by the property (P1), and strong Choquet by Theorem 10. Moreover, it is zero-dimensional since the class of ω-regular languages is closed under taking complements (see Theorem 3). It remains to apply Theorem 9. ◀

3.3 The other topologies

Proof of the main result. It is well known that $(\Sigma^{\mathbb{N}}, \tau_C)$ is metrizable and compact, and thus Polish.

- By Theorem 3.4 in [23], the implication (iii) $\Rightarrow$ (i), $\Delta^1_1(\Sigma^{\mathbb{N}})$ is a basis for a zero-dimensional Polish topology on $\Sigma^{\mathbb{N}}$. Recall that a Büchi Turing machine is **unambiguous** if every ω-word $\sigma \in \Sigma^{\mathbb{N}}$ has at most one accepting run. By Theorem 3.6 in [8], a subset of $\Sigma^{\mathbb{N}}$ is Δ^1_1 if and only if it is accepted by an unambiguous Büchi Turing machine. Therefore $\mathbb{B}_\delta = \Delta^1_1(\Sigma^{\mathbb{N}})$ is a basis for the zero-dimensional Polish topology τ_δ.
- Corollary 13 gives the result for the Büchi topology.
- Let (X, τ) be a Polish space, and $(C_n)_{n\in\mathbb{N}}$ be a sequence of closed subsets of (X, τ). By 13.2 in [16], the topology τ_n generated by $\tau \cup \{C_n\}$ is Polish. By 13.3 in [16], the topology τ_∞ generated by $\bigcup_{n\in\mathbb{N}} \tau_n$ is Polish. Thus the topology generated by $\tau \cup \{C_n \mid n \in \mathbb{N}\}$, which is τ_∞, is Polish. This shows that the automatic topology and the alphabetic topology are Polish since they refine the usual product topology on $\Sigma^{\mathbb{N}}$.
- Note that $S_{w,A}$ is a G_δ subset of $\Sigma^{\mathbb{N}}$, and thus a Polish subspace of $\Sigma^{\mathbb{N}}$, by 3.11 in [16]. Note also that $\Sigma^{\mathbb{N}} = \bigcup_{\emptyset\neq A\subseteq\Sigma} S_{\emptyset,A} \cup \bigcup_{\emptyset\neq A\subsetneq\Sigma, w\in\Sigma^*, b\in\Sigma\setminus A} S_{wb,A}$, and that this union is disjoint. As all the sets in this union are open for the strong alphabetic topology, they are also clopen for this topology. This shows that this topology is the countable sum of its restrictions to the sets in this union. As the strong alphabetic topology coincides with the usual topology on each of these sets, it is Polish, by 3.3 in [16]. ◀

4 Consequences for our topologies

4.1 Consequences not directly related to the polishness, concerning isolated points

Notation. If $z \in \{C, \delta, B, A, \alpha, s\}$, then the space $(\Sigma^{\mathbb{N}}, \tau_z)$ is denoted $\mathcal{S}_z$. The set of ultimately periodic ω-words on Σ is denoted $\mathrm{Ult} := \{u \cdot v^\omega \mid u, v \in \Sigma^* \setminus \{\emptyset\}\}$, and $P := \Sigma^{\mathbb{N}} \setminus \mathrm{Ult}$.

1. As noted in [26], Ult is the set of *isolated points* of $\mathcal{S}_B$ and $\mathcal{S}_A$ (recall that a point $\sigma \in \Sigma^{\mathbb{N}}$ is isolated if $\{\sigma\}$ is an open set). Indeed, each singleton $\{u\cdot v^\omega\}$ formed by an ultimately periodic ω-word is an ω-regular language, and thus each ultimately periodic ω-word is an isolated point of $\mathcal{S}_A$. Conversely, if $\{\sigma\}$ is τ_B-open, then it is ω-regular and then the ω-word σ is ultimately periodic (because any countable ω-regular language contains only ultimately periodic ω-words, see [1, 25, 30]).
2. Every nonempty ω-regular set contains an ultimately periodic ω-word, [1, 25, 30]. In particular, the set Ult of isolated points of $\mathcal{S}_B$ and $\mathcal{S}_A$ is *dense* , and a subset of $\mathcal{S}_B$ or $\mathcal{S}_A$ is dense if and only if it contains Ult.
3. Let X be a topological space in which the set $\mathbb{I}$ of isolated points is dense, for example $\mathcal{S}_B$ by (2).
   (a) A subset of X is nowhere dense (i.e., its closure has empty interior) if and only if it is *meager* (i.e., it is a countable union of nowhere dense sets). indeed, a meager set does not meet $\mathbb{I}$.
   (b) Recall that a subset L of a topological space Y has the *Baire property* if there is an open subset O of Y such that the symmetric difference $L\Delta O := (L\setminus O) \cup (O\setminus L)$ is meager. This is equivalent to say that $L = G \cup M$, where G is G_δ (i.e. a countable intersection of open sets) and M is meager (see 8.23 in [16]). Every subset of X has the Baire property (which is very uncommon, see for exemple 8.24 in [16]). Indeed,

we can even say that every subset of X can be written $L = O \cup N$, where O is open and N is nowhere dense (O is $L \cap \mathbb{I}$, and N is $L \setminus \mathbb{I}$).

(c) Recall that a topological space is a *Baire space* if the intersection of countably many dense open sets is dense. By the Baire category theorem, every completely metrizable space is Baire. The space X is Baire in a very strong way: the intersection of any family of dense sets is dense, since a subset of X is dense if and only if it contains $\mathbb{I}$. This is very unusual, since for example we can find two disjoint countable dense sets in $\mathbb{R}$.

(d) A consequence of (b), (c) and 8.38 in [16] is that any map from X into a second countable Polish space is *continuous on a dense* G_δ subset of X. Note that if X is $\mathcal{S}_B$ or $\mathcal{S}_A$, then any map from X into a topological space is continuous on the dense open set Ult of ultimately periodic ω-words.

(e) There is a strong version of the *Kuratowski-Ulam theorem* (see 8.41 in [16]): assume that X and Y are topological spaces in which the set of isolated points is dense. Then this property also holds in the product $X \times Y$, so that a subset of $X \times Y$ is meager if and only if it contains no isolated point, which is also equivalent to the fact that for each isolated point $x \in X$ (resp., $y \in Y$), the vertical (resp., horizontal) section at x (resp., y) contains no isolated point. An interesting example is the case of an infinitary rational relation $R(\mathcal{A}) \subseteq \Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ accepted by a 2-tape Büchi automaton $\mathcal{A}$ which may be synchronous or asynchronous. Indeed if $\Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ is equipped with the product topology of the Büchi topologies on $\Sigma^{\mathbb{N}}$ and $\Gamma^{\mathbb{N}}$ then a non-empty rational relation is always non-meager. This follows easily from the fact that $\mathrm{Dom}\big(R(\mathcal{A})\big) = \{x \in \Sigma^{\mathbb{N}} \mid \exists y \in \Gamma^{\mathbb{N}} \ \ (x, y) \in R(\mathcal{A})\}$ is a regular ω-language. Thus $\mathrm{Dom}\big(R(\mathcal{A})\big)$ contains an ultimately periodic word x and then $\{y \in \Gamma^{\mathbb{N}} \mid (x, y) \in R(\mathcal{A})\}$ is also a non-empty regular ω-language and so it contains also an ultimately periodic word y.

4.2 Consequences of the polishness

4.2.1 Consequences related to Cantor-Bendixson Theorem

We consider our topologies on $\Sigma^{\mathbb{N}}$, where Σ is a finite set with at least two elements. We give a (non exhaustive) list of results. We often refer to [16] when classical descriptive set theory is involved. The reader may read this book to see many other results.

4. (a) The union $P \cup \mathrm{Ult}$ is the *Cantor-Bendixson decomposition* of $\mathcal{S}_B$ and $\mathcal{S}_A$ (see 6.4 in [16]). This means that P is perfect (i.e., closed without isolated points) and Ult is countable open. Let us check that P is perfect. We argue by contradiction, so that we can find $\sigma \in P$ and an ω-regular language L such that $\{\sigma\} = L \setminus \mathrm{Ult}$. Note that $L \subseteq \{\sigma\} \cup \mathrm{Ult}$ is countable. But a countable regular ω-language contains only ultimately periodic words (see [25]), and thus $L \subseteq \mathrm{Ult}$, which is absurd.

(b) The closed subspace (P, τ_B) of $\mathcal{S}_B$ is *homeomorphic to the Baire space* $\mathbb{N}^{\mathbb{N}}$. Indeed, it is not empty since Ult is countable and $\Sigma^{\mathbb{N}}$ is not, zero-dimensional and Polish as a closed subspace of the zero-dimensional Polish space $\mathcal{S}_B$. By 7.7 in [16], it is enough to prove that every compact subset of (P, τ_B) has empty interior. We argue by contradiction, which gives a compact set K. Note that there is an ω-regular language L such that $P \cap L$ is a nonempty compact subset of K, so that we may assume that $K = P \cap L$. Theorem 2 gives $(U_i)_{i<n}$ and $(V_i)_{i<n}$ with $L = \bigcup_{i<n} U_i \cdot V_i^\omega$. On the other hand, L is not countable since every countable regular ω-language contains only ultimately periodic words and $K = P \cap L$ is non-empty. Thus $n > 0$ and, for

example, $U_0V_0^\omega$ is not countable.
This implies that we can find $v_0, v_1 \in V_0$ which are not powers of the same word. Indeed, we argue by contradiction. Let $v_0 \in V_0 \setminus \{\emptyset\}$, and $v \in \Sigma^*$ of minimal length such that v_0 is a power of v. We can find $w_0, w_1, ... \in V_0 \setminus \{\emptyset\}$ such that $\sigma := w_0w_1... \neq v^\omega$. Fix a natural number i. Then w_i and v_0 are powers of the same word w. By Corollary 6.2.5 in [22], v and w are powers of the same word u, and v_0 too. By minimality, $u = v$, and w_i is a power of v. Thus $\sigma = v^\omega$, which is absurd.
Let $u_0 \in U_0$, and $L' := \{u_0\}\cdot\{v_0, v_1\}^\omega$. Note that $P \cap L'$ is a τ_B-closed subset of K, so that it is τ_B-compact. As the identity map from $(P \cap L', \tau_B)$ onto $(P \cap L', \tau_C)$, (where τ_C is the Cantor topology), is continuous, $P \cap L'$ is also τ_C-compact. But the map $\alpha \mapsto u_0v_{\alpha(0)}v_{\alpha(1)}\dots$ is a homeomorphism from the Cantor space onto L', by Corollaries 6.2.5 and 6.2.6 in [22]. Thus $P \cap L' = L' \setminus \mathrm{Ult}$ is a dense closed subset of L'. Thus $P \cap L' = L'$, which is absurd since $u_0 \cdot v_0^\omega \in L' \setminus P$.

(c) By (b) and 7.9 in [16], for each Polish space Y, we can find a τ_B-closed set $F \subseteq P$ (an intersection of ω-regular sets) and a continuous bijection from F onto Y. In particular, if Y is not empty, then there is a *continuous surjection* from P onto Y extending the previous bijection. By 7.15 in [16], if moreover Y is perfect, then there is a continuous bijection from P onto Y.

(d) A consequence of (b), Corollary 13, and 7.10 in [16], is that the space $\mathcal{S}_B$ **is not** K_σ (i.e., countable union of compact sets).

4.2.2 Consequences related to Borel sets

5. (a) The *σ-algebra* $\mathcal{A}_z$ generated by $\mathbb{B}_z$ is exactly the σ-algebra of Borel subsets of $\mathcal{S}_C$ (generated by the open subsets of $\mathcal{S}_C$). Indeed, the identity map from $\mathcal{S}_z$ onto $\mathcal{S}_C$ is a continuous bijection. By 15.2 in [16], the inverse map is Borel, so that the two Borel structures coincide. It remains to note that the σ-algebra generated by $\mathbb{B}_z$ is exactly the σ-algebra of Borel subsets of $\mathcal{S}_z$.

(b) We can define a natural *hierarchy* in $\mathcal{A}_z$: let $\Sigma^z_1 := (\mathbb{B}_z)_\sigma$ be the set of countable unions of elements of $\mathbb{B}_z$, and, inductively on $1 \leq \xi < \omega_1$, $\Pi^z_\xi := \{\neg L \mid L \in \Sigma^z_\xi\}$ and $\Sigma^z_\xi := (\bigcup_{\eta<\xi} \Pi^z_\eta)_\sigma$. This hierarchy is actually the Borel hierarchy in the family of Borel subsets of $\mathcal{S}_z$, in the sense that $\Sigma^z_\xi = \mathbf{\Sigma}^0_\xi(\mathcal{S}_z)$ and $\Pi^z_\xi = \mathbf{\Pi}^0_\xi(\mathcal{S}_z)$ (see 11.B in [16]; we will also consider $\Delta^z_\xi := \Sigma^z_\xi \cap \Pi^z_\xi = \mathbf{\Delta}^0_\xi(\mathcal{S}_z)$ and $\mathbf{\Pi}^0_0(\mathcal{S}_z) := \mathbb{B}_z$). By the main result and 22.4 in [16], this hierarchy is strict and of length ω_1 (the first uncountable ordinal). This comes from the existence of universal sets for these classes (see 22.3 in [16]).
Recall that a subset $\mathcal{U}$ of $2^\mathbb{N} \times \Sigma^\mathbb{N}$ is **universal** for a class Γ of subsets of $\Sigma^\mathbb{N}$ if it is in Γ and the Γ subsets of $\Sigma^\mathbb{N}$ coincide with the vertical sections $\mathcal{U}_x = \{y \in \Sigma^\mathbb{N} \mid (x, y) \in \mathcal{U}\}$ of $\mathcal{U}$.
Note that a set is Borel for τ_C if and only if it is Borel for any topology τ_z, but the levels of the Borel hierarchy may differ for the two topologies. For instance any singleton associated with an ultimately constant word is not open for the Cantor topology and is actually open for the δ, Büchi, automatic, alphabetic and strong alphabetic topologies. Therefore the existence of universal sets for each level of the Borel hierarchy of $\mathcal{S}_z$ is different from the existence of universal sets for each level of the Borel hierarchy of $\mathcal{S}_C$ if $z \neq C$, and could lead to other consequences or new applications to other domains of theoretical computer science.

(c) The *Wadge theorem* holds (see 22.10 in [16]): let $z \in \{\delta, B\}$, $\xi \geq 1$ be a countable ordinal. A subset L of $\Sigma^\mathbb{N}$ is in $\Sigma^z_\xi \setminus \Pi^z_\xi$ if and only if it is in Σ^z_ξ, and for each

zero-dimensional Polish space X and any $\mathbf{\Sigma}^0_\xi$ subset B of X there is $f : X \to \mathcal{S}_z$ continuous with $B = f^{-1}(L)$ (we can exchange Σ^z_ξ and Π^z_ξ).

(d) The *Saint Raymond theorem* holds (see 35.45 in [16]): let Σ, Γ be finite sets with at least two elements, $L \subseteq \Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ be in $\mathcal{A}_z$ with Σ^z_2 vertical sections. Then $L = \bigcup_{n\in\mathbb{N}} L_n$, where L_n is in $\mathcal{A}_z$ and has Π^z_1 vertical sections.

(e) By 4.(b) and 13.7 in [16], for each L in $\mathcal{A}_B$, we can find a τ_B-closed set $F \subseteq P$ (an intersection of ω-regular sets) and a continuous bijection from F onto L. In particular, if L is not empty, then there is a *continuous surjection* from P onto L extending the previous bijection.

4.2.3 Consequences related to analytic sets

6. Recall that the analytic sets are the projections of the elements of $\mathcal{A}_z$. We saw in (5).(a) that τ_z and τ_C have the same Borel sets. This implies that τ_z and τ_C have the same *analytic* sets. Then the following items (a)-(d) are not new, but we cite them as interesting results about the topology τ_z.

(a) The class of analytic sets contains strictly A_z because of the existence of universal sets (see 14.2 in [16]).

(b) The *Lusin separation theorem* holds (see 14.7 in [16]): assume that $L, M \subseteq \Sigma^{\mathbb{N}}$ are disjoint analytic sets. Then there is S in $\mathcal{A}_z$ such that $L \subseteq S \subseteq \neg M$ (we say that S *separates* L from M).

(c) The *Souslin theorem* holds (see 14.11 in [16]): the elements of $\mathcal{A}_z$ are exactly the analytic sets whose complement is also analytic.

(d) By 14.13 in [16], an analytic set L has the *perfect set property* : either L is countable, or L contains a copy of $\mathcal{S}_C$ (this copy has size continuum and is compact).

(e) The *parametrization theorem for Borel sets* holds (see 35.5 in [16]). We say that a set is *co-analytic* if its complement is analytic. We can find a co-analytic subset $\mathcal{D}$ of $2^{\mathbb{N}}$, an analytic subset $\mathcal{S}$ of $2^{\mathbb{N}} \times \Sigma^{\mathbb{N}}$, and a co-analytic subset $\mathcal{P}$ of $2^{\mathbb{N}} \times \Sigma^{\mathbb{N}}$ such that the vertical sections $\mathcal{S}_\alpha$ and $\mathcal{P}_\alpha$ of $\mathcal{S}$ and $\mathcal{P}$ at any point α of $\mathcal{D}$ are equal, and the elements of $\mathcal{A}_z$ are exactly the sets $\mathcal{S}_\alpha$ for some α in $\mathcal{D}$.

4.2.4 Consequences related to uniformization problems

7. Let X, Y be sets, $L \subseteq X \times Y$, and $f : X \to Y$ be a partial map. We say that f *uniformizes* L if the domain of f is $\pi_0[L]$ and the graph of f is contained in L. In the sequel, Σ and Γ will be finite sets with at least two elements.

(a) Recall that if $L \subseteq \Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ is an infinitary rational relation, then L can be uniformized by a function whose graph is ω-rational (see Theorem 5 in [2]). Moreover, the inverse image of an ω-regular language by such a function is itself ω-regular, and thus the function is continuous for τ_B.

(b) The *Arsenin-Kunugui theorem* holds (see 18.18 in [16]): if $L \subseteq \Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ is in $\mathcal{A}_z$ and has K_σ vertical sections, then L can be uniformized by a function whose graph is in $\mathcal{A}_z$ (a K_σ set is a countable union of compact sets).

(c) Let $\xi \geq 1$ be a countable ordinal. The class $\mathbf{\Sigma}^0_\xi$ has the *number uniformization property* : if X is a zero-dimensional Polish space and $L \subseteq X \times \mathbb{N}$ is $\mathbf{\Sigma}^0_\xi$, then L can be uniformized by a function whose graph is $\mathbf{\Sigma}^0_\xi$. Consequently, for $z \in \{\delta, B\}$,

- Σ^z_ξ has the *generalized reduction property* : if $(L_n)_{n\in\mathbb{N}}$ is a sequence of Σ^z_ξ subsets of $\Sigma^{\mathbb{N}}$, then there is a sequence $(B_n)_{n\in\mathbb{N}}$ of pairwise disjoint Σ^z_ξ subsets of $\Sigma^{\mathbb{N}}$ with $\bigcup_{n\in\mathbb{N}} B_n = \bigcup_{n\in\mathbb{N}} L_n$ and $B_n \subseteq L_n$,

- Π^z_ξ has the *generalized separation property* : if $(L_n)_{n\in\mathbb{N}}$ is a sequence of Π^z_ξ subsets of $\Sigma^{\mathbb{N}}$ with empty intersection, then there is a sequence $(B_n)_{n\in\mathbb{N}}$ of Δ^z_ξ subsets of $\Sigma^{\mathbb{N}}$ with empty intersection and $B_n \supseteq L_n$ (see 22.16 in [16]).

Moreover,

- the class of co-analytic sets has the number uniformization property and the generalized reduction property,
- the class of analytic sets has the generalized separation property (see 35.1 in [16]).

(d) The *Novikov-Kondo theorem* holds (see 36.14 in [16]): if $L \subseteq \Sigma^{\mathbb{N}} \times \Gamma^{\mathbb{N}}$ is co-analytic, then L can be uniformized by a function whose graph is co-analytic.

5 Concluding Remarks

We have obtained in this paper new links and interactions between descriptive set theory and theoretical computer science, showing that four topologies considered in [26] are Polish, and providing many consequences of these results.

Notice that this paper is also motivated by the fact that the Gandy-Harrington topology, generated by the effective analytic subsets of a recursively presented Polish space, is an extremely powerful tool in descriptive set theory. In particular, this topology is used to prove some results of classical type (without reference to effective descriptive set theory in their statement). Among these results, let us mention the dichotomy theorems in [13, 17, 19, 20]. Sometimes, no other proof is known. Part of the power of this technique comes from the nice closure properties of the class Σ^1_1 of effective analytic sets (in particular the closure under projections).

The class of ω-regular languages has even stronger closure properties. So our hope is that the study of the Büchi topology, generated by the ω-regular languages, will help to prove some automatic versions of known descriptive results in the context of theoretical computer science.

From the main result, we know that there is a complete **distance** which is compatible with τ_z. It would be interesting to have a natural complete distance compatible with τ_B. We leave this as an open question for further study.

References

1 J. R. Büchi. On a decision method in restricted second order arithmetic. In Stanford University Press, editor, *Proceedings of the 1960 International Congress on Logic Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

2 C. Choffrut and S. Grigorieff. Uniformization of rational relations. In Juhani Karhumäki, Hermann A. Maurer, Gheorghe Paun, and Grzegorz Rozenberg, editors, *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 59–71. Springer, 1999.

3 R. S. Cohen and A. Y. Gold. ω-computations on Turing machines. *Theoretical Computer Science*, 6:1–23, 1978.

4 V. Diekert and M. Kufleitner. Fragments of first-order logic over infinite words. In *S. Albers and J.-Y. Marion, eds., STACS 2009, Freiburg, Germany, February 26-28, 2009, Proceedings*, pages 325–336, 2009. Online proceedings at DROPS and HAL, 2009.

5 J. Duparc, O. Finkel, and J.-P. Ressayre. The Wadge hierarchy of Petri nets ω-languages. In V. Brattka, H. Diener, and D. Spreen, editors, *Logic, Computation, Hierarchies*, volume 4 of *Ontos Mathematical Logic, collection of papers published in Honor of Victor Selivanov at the occasion of his sixtieth birthday*, pages 109–138. Ontos-Verlag, 2014.

6 O. Finkel. Borel ranks and Wadge degrees of omega context free languages. *Mathematical Structures in Computer Science*, 16(5):813–840, 2006.

7 O. Finkel. The complexity of infinite computations in models of set theory. *Logical Methods in Computer Science*, 5(4:4):1–19, 2009.

8 O. Finkel. Ambiguity of ω-languages of Turing machines. *Logical Methods in Computer Science*, 10(3:12):1–18, 2014.

9 O. Finkel and D. Lecomte. Classical and effective descriptive complexities of ω-powers. *Annals of Pure and Applied Logic*, 160(2):163–191, 2009.

10 O. Finkel and D. Lecomte. Decision problems for Turing machines. *Information Processing Letters*, 109:1223–1226, 2009.

11 O. Finkel and P. Simonnet. Topology and ambiguity in omega context free languages. *Bulletin of the Belgian Mathematical Society*, 10(5):707–722, 2003.

12 E. Grädel, W. Thomas, and W. 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.

13 L. A. Harrington, A. S. Kechris, and A. Louveau. A Glimm-Effros dichotomy for Borel equivalence relations. *Journal of the American Mathematical Society*, 3:903–928, 1990.

14 S. Hoffmann, S. Schwarz, and L. Staiger. Shift-invariant topologies for the Cantor space X^ω. *Theoretical Computer Science*, 679:145–161, 2017.

15 S. Hoffmann and L. Staiger. Subword metrics for infinite words. In Frank Drewes, editor, *Implementation and Application of Automata – 20th International Conference, CIAA 2015, Umeå, Sweden, August 18-21, 2015, Proceedings*, volume 9223 of *Lecture Notes in Computer Science*, pages 165–175. Springer, 2015.

16 A. S. Kechris. *Classical descriptive set theory*. Springer-Verlag, New York, 1995.

17 A. S. Kechris, S. Solecki, and S. Todorcevic. Borel chromatic numbers. *Advances in Mathematics*, 141(1):1–44, 1999.

18 L. H. Landweber. Decision problems for ω-automata. *Mathematical Systems Theory*, 3(4):376–384, 1969.

19 D. Lecomte. A dichotomy characterizing analytic digraphs of uncountable Borel chromatic number in any dimension. *Transactions of the American Mathematical Society*, 361(8):4181–4193, 2009.

20 D. Lecomte. Potential Wadge classes. *Memoirs of the American Mathematical Society*, 221(1038):vi+83, 2013.

21 H. Lescow and W. Thomas. Logical specifications of infinite computations. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *A Decade of Concurrency*, volume 803 of *Lecture Notes in Computer Science*, pages 583–621. Springer, 1994.

22 M. Lothaire. *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2002.

23 A. Louveau. *Ensembles analytiques et boréliens dans les espaces produit*, volume 78. Astérisque (SMF), 1980.

24 Y. N. Moschovakis. *Descriptive set theory*, volume 155 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, second edition, 2009.

25 D. Perrin and J.-E. Pin. *Infinite words, automata, semigroups, logic and games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

26 S. Schwarz and L. Staiger. Topologies refining the Cantor topology on X^ω. In C. S. Calude and V. Sassone, editors, *Theoretical Computer Science – 6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 271–285. Springer, 2010.

27 V. L. Selivanov. Wadge degrees of ω-languages of deterministic Turing machines. *RAIRO-Theoretical Informatics and Applications*, 37(1):67–83, 2003.

28 V. L. Selivanov. Wadge reducibility and infinite computations. *Mathematics in Computer Science*, 2(1):5–36, 2008.

29 O. Serre. Games with winning conditions of high Borel complexity. *Theoretical Computer Science*, 350(2-3):345–372, 2006.

30 L. Staiger. ω-languages. In *Handbook of formal languages, Vol. 3*, pages 339–387. Springer, Berlin, 1997.

31 L. Staiger. On the power of reading the whole infinite input tape. *Grammars*, 2 (3):247–257, 1999.

32 K. Wagner. On ω-regular sets. *Information and Control*, 43(2):123–177, 1979.

Separating Functional Computation from Relations

Ulysse Gérard[1] and Dale Miller[2]

1 **Inria Saclay & LIX/École Polytechnique, Palaiseau, France**
2 **Inria Saclay & LIX/École Polytechnique, Palaiseau, France**

Abstract

The logical foundation of arithmetic generally starts with a quantificational logic over relations. Of course, one often wishes to have a formal treatment of functions within this setting. Both Hilbert and Church added choice operators (such as the epsilon operator) to logic in order to coerce relations that happen to encode functions into actual functions. Others have extended the term language with confluent term rewriting in order to encode functional computation as rewriting to a normal form. We take a different approach that does not extend the underlying logic with either choice principles or with an equality theory. Instead, we use the familiar two-phase construction of focused proofs and capture functional computation entirely within one of these phases. As a result, our logic remains purely relational even when it is computing functions.

1998 ACM Subject Classification F.4.1 Mathematical Logic: Proof theory, Mechanical theorem proving

Keywords and phrases focused proof systems, fixed points, computation and deduction

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.23

1 Introduction

The development of the logical foundations of arithmetic generally starts with the first-order logic of relations to which constructors for zero and successor have been added. Various axioms (such as Peano's axioms) are then added to that framework in order to define the natural numbers and various relations among them. Of course, it is often natural to think of some computations, such as say, the addition and multiplication of natural numbers, as being *functions* instead of relations.

A common way to introduce functions into the relational setting is to enhance the equality theory. For example, Troelstra in [32, Section I.3] presents an intuitionistic theory of arithmetic in which all primitive recursive functions are treated as black boxes and every one of their instances, for example $23 + 756 = 779$, is simply added as an equation. A modern and more structured version of this approach is that of the $\lambda\Pi$-*calculus modulo* framework proposed by Cousineau & Dowek [10]: in that framework, the dependently typed λ-calculus (a presentation of intuitionistic predicate logic) is extended with a rich set of terms and rewriting rules on them. When rewriting is confluent, it can be given a functional programming implementation: the Dedukti proof checker [3] is based on this hybrid approach to treating functions in a relational setting.

A predicate can, of course, encode a function. For example, assume that we have a $n+1$-ary ($n \geqslant 0$) predicate R for which we can prove that the first n arguments uniquely determine the value of its last argument. That is, assume that the following formula is provable (here, $\bar{x}$ denotes the list of variables $x_1, \ldots, x_n$):

$$\forall \bar{x}([\exists y.R(\bar{x}, y)] \wedge \forall y \forall z[R(\bar{x}, y) \supset R(\bar{x}, z) \supset y = z]).$$

In this situation, an n-ary function f_R exists such that $f_R(\bar{x}) = y$ if and only if $R(\bar{x}, y)$. In order to formally describe the function f_R, Hilbert [23] and Church [9] evoked *choice operators* such as ϵ and ι which (along with appropriate axioms) are able to take a singleton set and return the unique element in that set. For example, in Church's Simple Theory of Types [9], the expression $\lambda x_1 \ldots \lambda x_n \iota(\lambda y.R(x_1, \ldots, x_n, y))$ provides a definition of f_R.

In this paper, we take a different approach to separating functional computations from more general reasoning with relations. We shall not extend the equational theory beyond the minimal equality on terms and we shall not use choice principles.

Although our approach to separating functions from relations is novel, it does not need any new theoretical results: we simply make direct use of several recent results in proof theory. In particular, our paper follows the following outline.

1. We formulate a sequent calculus proof system for Heyting arithmetic where fixed points and term equality are *logical connectives*: that is, they are defined via their left- and right-introduction rules. This work builds on earlier work by McDowell & Miller [26] and Momigliano & Tiu [30].
2. We replace Gentzen's sequent proofs with *focused proof systems* as developed by Andreoli, Baelde, and Liang & Miller [2, 24, 5]. Such inference systems structure proofs into two *phases*: the *negative* phase organizes *don't-care nondeterminism* while the *positive* phase organizes *don't-know nondeterminism*. In this way, the construction of a negative phase (reading it as a mapping from its conclusion to its premises) determines a function and the construction of the positive phase determines a more general nondeterministic relation.
3. Since $\forall x[P(x) \supset Q(x)] \equiv \exists x[P(x) \wedge Q(x)]$ whenever predicate P denotes a singleton set, the resulting *ambiguity of polarity* makes it possible to position such singleton predicates always into the negative phase. As mentioned above, a suitable treatment of singleton sets allows for a direct treatment of functions.
4. We exploit focused proof systems in a second and different fashion. If we view proofs of propositional formulas as denoting typed terms, then the usual representation of terms as function-applied-to-arguments occurs when primitive types are polarized negatively. If we set the polarity of primitive types to positive, we can turn the structure of terms inside out, yielding a representation of terms similar to *administrative normal form* [12]. Such a term representation allows us to translate common arithmetic expressions using functions into appropriate sequences of relational expressions that compute those functions. This approach to term representation builds on the $\lambda\kappa$-term calculus of Brock-Nannestad, Guenot, & Gustafsson [7] which is closely related to the LJQ and LJQ′ proof systems of Herbelin [22] and Dyckhoff & Lengrand [11], respectively.
5. Finally, the resulting proof system provides a means to take the specification of a relation and use it directly to compute a function (something that is not available directly when applying choice operators).

These various steps lead to the systematic construction of a single, expressive proof system in which functional computation is abstracted away from quantificational logic.

2 The basics of focusing in quantificational intuitionistic logic

In this section, we present a proof system for an intuitionistic theory of first-order quantification in two parts: Section 2.1 presents a proof system for the propositional fragment and Section 2.2 introduces quantification and equality of terms (at all types).

Structural rules

$$\frac{\Gamma, N \Downarrow N \vdash \cdot \Downarrow E}{\Gamma, N \Uparrow \cdot \vdash \cdot \Uparrow E}\, D_l \qquad \frac{C, \Gamma \Uparrow \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow C, \Theta \vdash \Delta_1 \Uparrow \Delta_2}\, S_l \qquad \frac{\Gamma \Uparrow P \vdash \cdot \Uparrow E}{\Gamma \Downarrow P \vdash \cdot \Downarrow E}\, R_l$$

$$\frac{\Gamma \Downarrow \cdot \vdash P \Downarrow \cdot}{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow P}\, D_r \qquad \frac{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow E}{\Gamma \Uparrow \cdot \vdash E \Uparrow \cdot}\, S_r \qquad \frac{\Gamma \Uparrow \cdot \vdash N \Uparrow \cdot}{\Gamma \Downarrow \cdot \vdash N \Downarrow \cdot}\, R_r$$

Negative phase introduction rules

$$\frac{\Gamma \Uparrow \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow t^+, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Gamma \Uparrow \cdot \vdash B_1 \Uparrow \cdot \quad \Gamma \Uparrow \cdot \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \wedge^- B_2 \Uparrow \cdot} \qquad \frac{}{\Gamma \Uparrow \cdot \vdash t^- \Uparrow \cdot} \qquad \frac{\Gamma \Uparrow f^+, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{}$$

$$\frac{\Gamma \Uparrow B_1, B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \wedge^+ B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Gamma \Uparrow B_1, \Theta \vdash \Delta_1 \Uparrow \Delta_2 \quad \Gamma \Uparrow B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Gamma \Uparrow B_1 \vee B_2, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Gamma \Uparrow B_1 \vdash B_2 \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash B_1 \supset B_2 \Uparrow \cdot}$$

Positive phase introduction rules

$$\frac{\Gamma \Downarrow \cdot \vdash B_1 \Downarrow \cdot \qquad \Gamma \Downarrow B_2 \vdash \cdot \Downarrow E}{\Gamma \Downarrow B_1 \supset B_2 \vdash \cdot \Downarrow E} \qquad \frac{}{\Gamma \Downarrow \cdot \vdash t^+ \Downarrow \cdot} \qquad \frac{\Gamma \Downarrow \cdot \vdash B_1 \Downarrow \cdot \qquad \Gamma \Downarrow \cdot \vdash B_2 \Downarrow \cdot}{\Gamma \Downarrow \cdot \vdash B_1 \wedge^+ B_2 \Downarrow \cdot}$$

$$\frac{\Gamma \Downarrow \cdot \vdash B_i \Downarrow \cdot}{\Gamma \Downarrow \cdot \vdash B_1 \vee B_2 \Downarrow \cdot}\, i \in \{1, 2\} \qquad \frac{\Gamma \Downarrow B_i \vdash \cdot \Downarrow E}{\Gamma \Downarrow B_1 \wedge^- B_2 \vdash \cdot \Downarrow E}\, i \in \{1, 2\}$$

Figure 1 The propositional fragment of cut-free LJF.

2.1 Propositional intuitionistic logic

In this section, we present propositional intuitionistic logic and a focused proof system for it. Propositional intuitionistic logic formulas are given by the logical connectives $\wedge$, $\vee$, and $\supset$, the logical constants t and f, and atomic formulas. The focused system in Figure 1 contains not formulas but *polarized* formulas. Such polarized formulas differ from unpolarized formulas in two ways. First, the conjunction is replaced with two conjunctions $\wedge^+$ and $\wedge^-$ and the unit of conjunction t with t^+ and t^-. Second, every atomic formula A is assigned either a positive or negative polarity in an arbitrary but fixed fashion. Thus, one can fix the polarity of atomic formulas (propositional variables) such that they are all positive or all negative or some mixture of positive and negative. A polarized formula is *positive* if it is a positive atomic formula or its top-level logical connective is either t^+, f, $\wedge^+$, or $\vee$. A polarized formula is *negative* if it is a negative atomic formula or its top-level logical connective is either t^-, $\wedge^-$, or $\supset$.

Figure 1 contains the structural and introduction rules for the propositional fragment of the LJF focused proof system [24]. That proof system uses the following two kinds of sequents: *unfocused* sequents have the form $\Gamma \Uparrow \Theta \vdash \Delta_1 \Uparrow \Delta_2$, while *focused* sequents have the form $\Gamma \Downarrow \Theta \vdash \Delta_1 \Downarrow \Delta_2$. In those inference rules, the syntactic variables Δ, Θ, and Γ (possibly with subscripts) range over multisets of polarized formulas; P denotes a positive formula; N denotes a negative formula; C denotes either a negative formula or a positive atom; E denotes either a positive formula or a negative atom; and B denotes any polarized formula. Since we are working with an intuitionistic sequent system, we require that all sequents in a focused proof have exactly one formula on the right: that is, the multiset union of Δ_1 and Δ_2 is a singleton. Since we are considering only *single-focused* proof systems (as opposed to *multifocused* proof systems [8]), we also require that sequents of the form $\Gamma \Downarrow \Theta \vdash \Delta_1 \Downarrow \Delta_2$ have the property that the multiset union of Θ and Δ_1 be always a singleton. An invariant

in the construction of LJF proofs is that Γ will be a multiset that can contain only negative formulas and positive atoms. Every sequent in LJF denotes a standard sequent in LJ: simply replace $\Uparrow$ and $\Downarrow$ with commas. An unfocused sequent of the form $\Gamma \Uparrow \cdot \vdash \cdot \Uparrow E$ is also called a *border* sequent.

A *derivation* is a tree structure of occurrences of inference rules: a derivation has one conclusion (the endsequent) and possibly several premises. A derivation with no premises is a (focused) proof. A derivation that contains only negative sequents is a *negative phase*: such a phase contains introduction rules for negative connectives, and the storage rules (S_l and S_r). A derivation that contains only positive sequents is a *positive phase*: such a phase contains introduction rules for positive connectives. A *bipole* is a derivation whose conclusion and premises are all border sequents: also, when reading the inference rules from the bottom up, the first inference rule is a decide rule (either D_l or D_r); the next rules are positive introduction rules; then there is a release rule (either R_l or R_r); followed by negative introduction rules and storage rules (either S_l or S_r). In other words, a bipole is the joining of a single positive phase to possibly several negative phases.

Figure 1 contains neither the initial rule nor the cut rule. Although the cut rule and the cut-elimination theorem play important roles in justifying the design of focused proof systems, they play a minor role in this paper (for example, cut-elimination is not part of our notion of computation). The initial rule will be important but not globally: we introduce it later when we need (variants of) it.

2.2 Quantification and term equality

In order to treat first-order quantification, sequents are extended to permit the proof-level binding mechanism of *eigenvariables* [16]. To that end, we prefix all $\Uparrow$ and $\Downarrow$ sequents with $\Sigma :$, where Σ is a list of variables that are considered bound over the sequent. When we write a prefix as $y : \tau, \Sigma$, we imply that y does not appear as one of the variables in Σ. The inference rules for term equality and quantification are displayed in Figure 2 and are taken from early papers by Schroeder-Heister [28] and Girard [18]: see also [26]. Formulas with a top-level $\forall$ have negative polarity while formulas with a top-level $\exists$ or equality have positive polarity. The expression $[t/x]B$ denotes the $\beta\eta$-long normal form of $(\lambda x.B)t$ and the judgment $\Sigma \vdash t : \tau$ denotes the fact that t is a term in $\beta\eta$-long form and with type τ. The typing judgment will be made more precise and generalized later in Section 6.

While provability in the propositional fragment is known to be decidable [16], it has been shown in [33] that adding these rules for term equality and quantification results in an undecidable logic even if we restrict to just first-order terms and quantifiers and even without any predicate symbols (and, hence, without atomic formulas).

3 Inference rules for the fixed point connective

We shall now add to our collection of logical connectives a fixed point operator. There have been many treatments of fixed points and induction within proof systems such as those involving Peano's axioms and induction schemes or those using a specially designed proof system such as Scott induction [19]. Here, we restrict our attention to the rather minimalistic setting where the fixed point operator μ is treated as a logical connective in the sense that it has left- and right-introduction rules: these rules simply unfold μ-expressions. While the resulting fixed point operator is self-dual and rather weak, it can still play a useful role in proving some weak theorems of arithmetic [18, 28, 26] and it can provide an interesting proof theory for aspects of model checking [4, 20, 31]. It is possible to describe a more powerful

Typed first-order quantification rules

$$\frac{\Sigma \vdash t : \tau \qquad \Sigma : \Gamma \Downarrow [t/x]B \vdash \cdot \Downarrow E}{\Sigma : \Gamma \Downarrow \forall x_\tau.B \vdash \cdot \Downarrow E} \qquad \frac{y : \tau, \Sigma : \Gamma \Uparrow \cdot \vdash [y/x]B \Uparrow \cdot}{\Sigma : \Gamma \Uparrow \cdot \vdash \forall x_\tau.B \Uparrow \cdot}$$

$$\frac{y : \tau, \Sigma : \Gamma \Uparrow [y/x]B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Gamma \Uparrow \exists x_\tau.B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Sigma \Uparrow \cdot \vdash t : \tau \Uparrow \cdot \quad \Sigma : \Gamma \Downarrow \cdot \vdash [t/x]B \Downarrow \cdot}{\Sigma : \Gamma \Downarrow \cdot \vdash \exists x_\tau.B \Downarrow \cdot}$$

Equality rules

$$\frac{\Sigma\theta : \Gamma\theta \Uparrow \Theta\theta \vdash \Delta_1\theta \Uparrow \Delta_2\theta}{\Sigma : \Gamma \Uparrow s = t, \Theta \vdash \Delta_1 \Uparrow \Delta_2}\ \dagger \qquad \frac{}{\Sigma : \Gamma \Uparrow s = t, \Theta \vdash \Delta_1 \Uparrow \Delta_2}\ \ddagger \qquad \frac{}{\Sigma : \Gamma \Downarrow \cdot \vdash t = t \Downarrow \cdot}$$

There are two provisos: (†) θ is the mgu of s and t. (‡) t and s are not unifiable.

Figure 2 Focused proof rules for quantification and term equality.

proof system for fixed points that uses induction and co-induction rules to describe the introduction rules for the *least* and *greatest* fixed points [26, 30].

The logical constant μ is actually parameterized by a list of typed constants as follows:

$$\mu^n_{\tau_1,\ldots,\tau_n} : ((\tau_1 \to \cdots \to \tau_n \to o) \to \tau_1 \to \cdots \to \tau_n \to o) \to \tau_1 \to \cdots \to \tau_n \to o$$

where $n \geqslant 0$ and $\tau_1, \ldots, \tau_n$ are simple types. (Following Church [9], we use o to denote the type of formulas.) Expressions of the form $\mu^n_{\tau_1,\ldots,\tau_n} B t_1 \ldots t_n$ will be abbreviated as simply $\mu B \bar{t}$ (where $\bar{t}$ denotes the list of terms $t_1 \ldots t_n$). We shall also restrict fixed point expressions to use only *monotonic* higher-order abstraction: that is, in the expression $\mu^n_{\tau_1,\ldots,\tau_n} B t_1 \ldots t_n$ the expression B is equivalent (via $\beta\eta$-conversion) to $\lambda P_{\tau_1 \to \cdots \to \tau_n \to o}\, \lambda x^1_{\tau_1} \ldots \lambda x^n_{\tau_n}\, B'$ and where all occurrences of the variable P in B' occur to the left of an implication an even number of times. The unfolding of the fixed point expression $\mu B\, \bar{t}$ yields $B(\mu B)\, \bar{t}$ and the introduction rules for μ establish the logical equivalence of these two expressions.

▶ **Example 1.** Assume that we have a primitive type i and that there are two typed constants $z : i$ and $s : i \to i$. We shall abbreviate the terms z, $(s\ z)$, $(s\ (s\ z))$, $(s\ (s\ (s\ z)))$, etc by $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$, $\mathbf{3}$, etc. The following two named fixed point expressions define the natural number predicate and the ternary relation of addition.

$$\begin{aligned} \mathrm{nat} &= \mu\lambda N \lambda n (n = \mathbf{0} \vee \exists n' (n = s\ n' \wedge^+ N\ n')) \\ \mathrm{plus} &= \mu\lambda P \lambda n \lambda m \lambda p ((n = \mathbf{0} \wedge^+ m = p) \vee \exists n' \exists p' (n = s\ n' \wedge^+ p = s\ p' \wedge^+ P\ n'\ m\ p')) \end{aligned}$$

The following theorem, proved using induction, states that the plus relation describes a (total) functional dependency between its first two arguments and its third.

$$\forall m, n (\mathrm{nat}\ m \supset \exists k (\mathrm{plus}\ m\ n\ k)) \wedge \forall m, n, p, q (\mathrm{plus}\ m\ n\ p \supset \mathrm{plus}\ m\ n\ q \supset p = q)$$

3.1 Focusing and unfolding

The natural rules for unfolding μ-expressions are given as the first two inference rules of Figure 3. Here, we have assigned to such expressions the positive polarity. Since the left-introduction and right-introduction rules for μ-expressions are the same (i.e., they are unfolded), they could have been polarized negatively as well. If we were to add an induction rule in order to have μ-expressions capture *least* fixed points, the use of the positive polarity would be the most natural choice [27].

Focused sequent calculus proof systems were originally developed for quantificational logic – as opposed to arithmetic – and in that setting the bottom-up construction of the negative phase causes sequents to get strictly smaller (counting, for example, the number of occurrences of logical connectives). As a result, it was possible to design focused proof systems in which decide rules were not applied until *all* invertible rules were applied. We shall say that such proofs systems are *strongly* focused proof systems: examples of such systems can be found in [2, 24].

As is obvious from the first two inference rules in Figure 3, the size of the formulas in the negative phase can increase when μ-expressions are unfolded. Thus, a more flexible approach to building negative phases should be considered. Some focused proof systems have been designed in which a decide rule can be applied without consideration of whether all or some of the invertible rules have been applied. Following [29], such proof systems are called *weakly* focused proof systems: an early example of such a proof system is Girard's LC [17]. Since we wish to use the negative phase to do functional style, determinate computation, a weakly focused system – with its possibility to stop in many different configurations – cannot provide the foundations that we need.

Instead of strongly and weakly focused proof systems, we modify the notion of strongly focusing by allowing certain explicitly described μ-expressions appearing in the negative phase to be *suspended*. In that case, one can switch from a negative phase to a positive phase (using a decide rule) when the only remaining formulas in the negative phase are suspendable. In that case, those formulas are "put aside" during the processing of the positive phase and are reinstated when the positive phase switches to the negative phase (using a release rule). In more detail, let $\mathcal{S}$ denote a *suspension* predicate: this predicate is defined only on μ-expressions and if $\mathcal{S}$ holds for $(\mu B\bar{t})$ then we say that this expression is suspended. The $\mathtt{unfoldL}$ rule in Figure 3 has the proviso that $\mathcal{S}$ does not hold of the μ-expression that is the subject of that inference rule. In order to accommodate suspended formulas, ⇓-sequents need to contain a new multiset zone, denoted by the syntactic variable Ω: in particular, they now have the structure $\Gamma \Downarrow \Theta; \Omega \vdash \Delta_1 \Downarrow \Delta_2$. All positive introduction rules ignore this new zone: for example, the left-introduction of $\wedge^-$ will now be written as

$$\frac{\Gamma \Downarrow B_i; \Omega \vdash \cdot \Downarrow E}{\Gamma \Downarrow B_1 \wedge^- B_2; \Omega \vdash \cdot \Downarrow E}\ i \in \{1, 2\}.$$

The suspension property $\mathcal{S}$ is defined at the mathematics level and, as a result, can make use of syntactic details about μ-expressions. For example, this property could be defined to hold for a μ-expression that contains more than, say, 100 symbols or when the first term in the list $\bar{t}$ is an eigenvariable. However, in order to guarantee that the negative phase is determinate, we need to require the following property:

(∗) For all μ-expressions $(\mu B\bar{t})$ and for all substitutions θ defined on the eigenvariables free in that μ-expression, if $\mathcal{S}$ holds for $(\mu B\bar{t})\theta$ then $\mathcal{S}$ holds for $(\mu B\bar{t})$.

That is, if an instance of a μ-expression satisfies $\mathcal{S}$ after a substitution is applied, it must satisfy $\mathcal{S}$ before it was applied. This condition rules out the possible suspension condition "holds if it contains 100 symbols" but it allows the condition "holds if the first term in $\bar{t}$ is an eigenvariable". Furthermore, suspension properties should not, in general, be invariant under substitution since otherwise a suspended formula will remain suspended during the construction of a proof: it can only be used within the initial rule.

▶ **Example 2.** Consider the suspension predicate that is true of μ-expressions $\mu B\, t_1 \ldots t_n$ if and only if $n \geqslant 2$ and t_1 and t_2 are the same variable. Clearly, property (∗) does not hold

Fixed point rules

$$\frac{\Sigma : \Gamma \Uparrow B(\mu B)\bar{t}, \Theta \vdash \Delta \Uparrow E}{\Sigma : \Gamma \Uparrow \mu B\, \bar{t}, \Theta \vdash \Delta \Uparrow E}\ \mathtt{unfoldL}\dagger \qquad \frac{\Sigma : \Gamma \Downarrow \cdot \vdash B(\mu B)\bar{t} \Downarrow \cdot}{\Sigma : \Gamma \Downarrow \cdot \vdash \mu B\, \bar{t} \Downarrow \cdot}\ \mathtt{unfoldR}$$

Modified versions of the decide and release rules

$$\frac{\Sigma : \Gamma, N \Downarrow N; \Omega \vdash \cdot \Downarrow E}{\Sigma : \Gamma, N \Uparrow \Omega \vdash \cdot \Uparrow E}\ D_l\ddagger \qquad \frac{\Sigma : \Gamma \Downarrow \cdot; \Omega \vdash P \Downarrow \cdot}{\Sigma : \Gamma \Uparrow \Omega \vdash \cdot \Uparrow P}\ D_r\ddagger$$

$$\frac{\Sigma : \Gamma \Uparrow P, \Omega \vdash \cdot \Uparrow E}{\Sigma : \Gamma \Downarrow P; \Omega \vdash \cdot \Downarrow E}\ R_l \qquad \frac{\Sigma : \Gamma \Uparrow \Omega \vdash N \Uparrow \cdot}{\Sigma : \Gamma \Downarrow \cdot; \Omega \vdash N \Downarrow \cdot}\ R_r$$

Initial rule

$$\frac{P \in \Omega}{\Sigma : \Gamma \Downarrow \cdot; \Omega \vdash P \Downarrow \cdot}\ I_r$$

The proviso † requires that $\mu B\, \bar{t}$ does not satisfy $\mathcal{S}$. The proviso ‡ requires Ω to be a multiset of μ-expressions that satisfy $\mathcal{S}$.

Figure 3 Rules governing fixed point unfolding, suspensions, and initial sequents.

and the construction of the negative phase can be non-confluent. For example, let A be $\mu\lambda p\lambda x\lambda y.x = a$ (where a is a constant) and consider the sequent $\Gamma \Uparrow u = v, Auv \vdash \cdot \Uparrow (E\ u)$. Since Auv is a μ-expression for which $\mathcal{S}$ does not hold, unfolding is applicable and yields the sequent $\Gamma \Uparrow u = v, u = a \vdash \cdot \Uparrow (E\ u)$ which then leads to the border sequent $\Gamma \Uparrow \cdot \vdash \cdot \Uparrow (E\ a)$. However, the first step in the negative phase of the original sequent could have been the equality introduction, which yields $\Gamma \Uparrow Auu \vdash \cdot \Uparrow (E\ u)$ and this must mark the end of the negative phase since $A\ u\ u$ is a suspended formula.

Fortunately, this non-confluent behavior is ruled out by the $(*)$ property above. To see this, let $\mathcal{C}$ be an $\Uparrow$-sequent that and let Ξ be a negative phase that has $\mathcal{C}$ as its endsequent and with premises that are border sequents. If we collect the premises of Ξ into a set, say, $\mathcal{P}$, then we call $\mathcal{P}$ an *invertible decomposition* of $\mathcal{C}$. It is easy to show, via permutations of inference rules, that if $\mathcal{C}$ has $\mathcal{P}_1$ and $\mathcal{P}_2$ as invertible decompositions, then $\mathcal{P}_1 = \mathcal{P}_2$. The $(*)$ condition enables the permutation of the equality left-introduction rule and the $\mathtt{unfoldL}$ rule.

▶ **Definition 3** (Purely positive formula). A polarized formula in which all occurrences of logical connectives are polarized positively is called a *purely positive* formula. A μ-expression that is also purely positive will also be called a purely positive fixed point expression.

Horn clauses (Prolog) can provide immediate examples of purely positive fixed points as illustrated in Example 1. Let B be a purely positive formula. If $\Sigma : \Gamma \Downarrow \cdot \vdash B \Downarrow \cdot$ is provable then all proofs of that sequent are built of only positive right-introduction rules for t^+, $\wedge^+$, $\vee$, $\exists$, μ (unfolding) and equality. Similarly, if $\Sigma : \Gamma \Uparrow B \vdash \cdot \Uparrow \cdot$ is provable then all proofs of that sequent are built of only negative left-introduction rules for t^+, $\wedge^+$, $\vee$, $\exists$, μ (unfolding), and equality. Thus, focused proofs of B and $B \supset f^+$ are achieved by using only one phase. In particular, such proofs do not contain structural rules nor the initial rule. As a result, synthetic inference rules are not decidable since they can encode arbitrary Horn clause specifications.

3.2 Phases as abstractions

Focused proof systems make it possible to define new inference rules by abstracting away details used in the construction of phases. The positive phase allows a simple abstraction since there is exactly one formula under focus in a positive sequent. A positive phase can be seen as the (derived) inference rule with a conclusion that is a border sequent and with premises that are marked by release rules.

There are, however, at least two challenges to making abstractions of negative phases. First, the premises of a negative phase may repeat the same sequents many times since there can be many paths to compute the result of a function. We shall choose to denote as the collections of premises of the negative phase the *set* of border sequents (instead of as a *multiset*). Second, there are many ways to process the don't-care nondeterminism that is possible when applying invertible rules. We will abstract away from those differences by simply ignoring *how* a phase is constructed since all constructions yield the same border sequents.

This second abstraction flows from the same motivation used in confluent rewriting systems: once a path to a normal form is found, no other paths need to be considered since all other paths must yield the same normal form.

4 The polarity ambiguity of singleton sets

As we mentioned in the introduction, singleton sets can be used to help convert relations to functions: if the $n+1$-ary relation R describes a function from its first n arguments to its last argument then the expression $(\lambda y.R(x_1,\ldots,x_n,y))$ denotes a singleton set (given fixed values for $x_1,\ldots,x_n$). The choice operators ϵ or ι can then be applied to this singleton set to extract that element, resulting in a proper function $\lambda x_1 \ldots \lambda x_n \iota(\lambda y.R(x_1,\ldots,x_n,y))$.

Singleton sets play a role here as well. In fact, let P be a predicate of one argument so that it is provable that P is a singleton, namely,

$$(\exists x.P(x)) \wedge (\forall x, y.P(x) \supset P(y) \supset x = y)$$

As a consequence, the formulas $\exists x.P(x) \wedge Q(x)$ and $\forall x.P(x) \supset Q(x)$ are equivalent. If we used the ι-operator, these formulas would also be equivalent to $Q(\iota P)$.

Note that the sequent calculus treatments of $\exists x.P(x) \wedge Q(x)$ and $\forall x.P(x) \supset Q(x)$ are strikingly different. In particular, a proof of $\Sigma; \Gamma \Downarrow \cdot \vdash \exists x.P(x) \wedge Q(x) \Downarrow \cdot$ proceeds by guessing a term t and then attempting to prove $\Sigma; \Gamma \Downarrow \cdot \vdash P(t) \Downarrow \cdot$ and $\Sigma; \Gamma \Downarrow \cdot \vdash Q(t) \Downarrow \cdot$. Of course, since P denotes a singleton, there is at most one correct guess t and that guess is confirmed after it is inserted into the proof. On the other hand, a proof of $\Sigma; \Gamma \Uparrow \cdot \vdash \forall x.P(x) \supset Q(x) \Uparrow \cdot$ can be seen as computing the value that satisfies P. Proof construction for that sequent leads to proving $y, \Sigma : \Gamma \Uparrow P(y) \vdash Q(y) \Uparrow \cdot$. As mentioned in Section 3.1, this phase will move to completion by repeatedly unfolding fixed points and if the phase completes, the eigenvariable y will be instantiated to be the unique term t. Thus, the premises of this completed phase will have the shape $\Sigma; \Gamma \Uparrow \vdash \cdot \Uparrow Q(t)$ (assuming for the sake of argument that $Q(t)$ is a positive formula).

▶ **Example 4.** Using the definitions in Example 1, consider the construction of a negative phase of the form $x, \Sigma : \Gamma \Uparrow \text{plus } \mathbf{2}\ \mathbf{3}\ x \vdash \cdot \Uparrow (Q\ x)$ Since plus is a μ-expression, this sequent is proved by an $\mathtt{unfoldL}$ inference rule (assuming that S is false for all μ-expressions, i.e., nothing should be suspended). Unfolding yields an expression with a top-level disjunction, namely, $x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^+ \mathbf{3} = x) \vee \exists n' \exists x' (\mathbf{2} = s\ n' \wedge^+ x = s\ x' \wedge^+ \text{plus } n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x)$.

Following the left-introduction for that disjunction, we are left with proving two sequents: the left premises, $x, \Sigma : \Gamma \Uparrow ((\mathbf{2} = \mathbf{0} \wedge^+ \mathbf{3} = x) \vdash \cdot \Uparrow (Q\ x)$ is proved immediately since $\mathbf{2} = \mathbf{0}$ is not unifiable (Figure 2). A proof of the second premise must proceed as follows

$$\dfrac{\dfrac{x', \Sigma : \Gamma \Uparrow \text{plus}\ \mathbf{1}\ \mathbf{3}\ x' \vdash \cdot \Uparrow (Q\ (s\ x'))}{x, n', x', \Sigma : \Gamma \Uparrow (\mathbf{2} = s\ n' \wedge^+ x = s\ x' \wedge^+ \text{plus}\ n'\ \mathbf{3}\ x') \vdash \cdot \Uparrow (Q\ x)}}{x, \Sigma : \Gamma \Uparrow (\exists n' \exists x' (\mathbf{2} = s\ n' \wedge^+ x = s\ x' \wedge^+ \text{plus}\ n'\ \mathbf{3}\ x')) \vdash \cdot \Uparrow (Q\ x)}$$

(Here, the double line between sequents denotes the application of possibly several inference rules.) After several more inference steps, the negative phase terminates with the border premise $\Sigma; \Gamma \Uparrow \cdot \vdash \cdot \Uparrow (Q\ \mathbf{5})$. By ignoring the internal structure of phases, we have just the synthetic inference rule

$$\dfrac{\Sigma; \Gamma \Uparrow \cdot \vdash \cdot \Uparrow (Q\ \mathbf{5})}{x, \Sigma : \Gamma \Uparrow \text{plus}\ \mathbf{2}\ \mathbf{3}\ x \vdash \cdot \Uparrow (Q\ x)}\ .$$

Furthermore, there were no choices involved in computing this phase. Note that the actual specification of the relation plus is used to compute the addition as a function. Later in Section 6 we shall show how we can use that synthetic inference rule to capture the more familiar looking rule

$$\dfrac{\Sigma; \Gamma \Uparrow \cdot \vdash \cdot \Uparrow (Q\ \mathbf{5})}{\Sigma; \Gamma \Uparrow \cdot \vdash \cdot \Uparrow (Q\ (\mathbf{2} + \mathbf{3}))}\ .$$

▶ **Example 5.** Employing the suspension mechanism makes it possible for functional computation to be mixed with symbolic computation. For example, let multiplication be defined as the following fixed point expression.

$$\text{times} = \mu\lambda P \lambda n \lambda m \lambda p((n = \mathbf{0} \wedge^+ p = \mathbf{0}) \vee \exists n' \exists p' (n = s\ n' \wedge^+ P\ n'\ m\ p' \wedge^+ \text{plus}\ p'\ m\ p))$$

The theorem that states that $(0 \times (x + 1)) + y = y$ can be encoded and proved in this setting by taking two steps. First we translate this expression into the following sequent (using a technique described in Section 6):

$$y, \Sigma : \Gamma \Uparrow \cdot \vdash \forall u.\ \text{times}\ \mathbf{0}\ (s\ x)\ u \supset \forall v.\ \text{plus}\ u\ y\ v \supset v = y \Uparrow \cdot.$$

Here, we assume the (rather typical) suspension mechanism that classifies μ-expressions as suspendable if they are built from plus and times and their first argument is an eigenvariable. Thus, when this sequent is reduced to

$$u, v, y, \Sigma : \Gamma \Uparrow \text{times}\ \mathbf{0}\ (s\ x)\ u,\ \text{plus}\ u\ y\ v \vdash v = y \Uparrow \cdot,$$

only the times-expression can be unfolded. After that unfolding, the eigenvariable u will be instantiated and the plus-expression can then also be unfolded. Finally, the negative phase ends with the border sequent $y, \Sigma : \Gamma \Uparrow \cdot \vdash \cdot \Uparrow y = y$ which is proved by a D_r rule followed by the right-introduction rule for equality.

5 Equivalence classes

Equivalence relations play important roles in computation and reasoning. Occasionally, we have a relation that is not functional but all the possible outcomes are equivalent, for some specific equivalence relation. For example, if two lists are considered equivalent when they are

permutations of each other, then the equivalence class of lists modulo that relation encodes multisets. Similarly, if two pairs of integers (x, y) and (w, z) (where y and z are not zero) are considered equivalent when $xz = wy$ then equivalence classes encode rational numbers.

The ambiguity of singletons can be lifted to computation with equivalence classes in the following sense. Let ρ be an equivalence relation. The familiar notion $[x]_\rho$ for the ρ-equivalence class containing x is just syntactic sugar for $\lambda y.x\rho y$. (Define logical equivalence in the usual way: $A \equiv B$ is an abbreviation for $(A \supset B) \wedge (B \supset A)$.)

Assume that ρ is an equivalence relation and that the following holds for $Q : i \to o$.

$$\forall x \forall y.\ x\ \rho\ y \supset [Q(x) \equiv Q(y)]$$

(Note that this theorem is immediate for all $Q : i \to o$ when ρ is equality.) The following equivalence holds.

$$[\forall x \in [y]_\rho \supset Q(x)] \equiv [\exists x \in [y]_\rho \wedge Q(x)]$$

In a more informal mathematical notation, one might replace either the above existential or universal expression with $Q([y]_\rho)$. While we shall not use this expression (it involves a typing error), it conveys the usual mathematical sense of this ambiguity: if we show that one member of an equivalence class satisfies such a property Q then all members of that equivalence class satisfy Q.

Obviously, we can generalize the notion of functional dependency to the following

$$\forall \bar{x}([\exists y.R(\bar{x}, y)] \wedge \forall y \forall z[R(\bar{x}, y) \supset R(\bar{x}, z) \supset y \rho z]),$$

which states that the n-ary relation is a total function up to ρ. Thus, during the construction of a proof where one is asked to pick a term t that makes $R(x_1, \ldots, x_n, t)$ true, one can instead compute just any term t' such that $R(x_1, \ldots, x_n, t')$ (as long as the property established – Q above – is ρ-invariant). In that setting, we can also extend the phase-abstraction mechanism to exclude border premises that differ up to ρ.

6 Term representation: turning formulas inside-out

6.1 Term annotations for propositional LJF

In Section 2.2 we extended the proof system in Figure 1 with quantifiers and term structures and in Section 3 with recursive definitions. Here we extend that original proof system in two different directions. First, instead of having all predicates (such as nat, plus, and times) be defined, we consider the usual approach to propositional logic where formulas can contain *undefined* atoms. When such atoms appear in polarized formulas, atomic formulas must be provided with an arbitrary but fixed polarity. Following the design of *LJF* [24], we extend the proof system in Figure 1 by adding the two variants of the initial rule displayed on the right. Here, N_a ranges over negatively polarized atoms and P_a ranges over positively polarized atoms. Given that we are working with a propositional logic, it is possible to use a strongly focused version of LJF (as was given in [24]) and to insist that all formulas in the negative phase are processed in a left-to-right discipline. As a result, it is possible to fuse the store-left rule (S_l) with other rules.

$$\frac{}{\Gamma \Downarrow N_a \vdash \cdot \Downarrow N_a}\ I_l \qquad \frac{}{\Gamma, P_a \Downarrow \cdot \vdash P_a \Downarrow \cdot}\ I_r$$

The completeness theorem for *LJF* can be stated as follows. Given an (unpolarized) formula B, a *polarization* of B is a formula that results from replacing every occurrence in B of $\wedge$ with either $\wedge^+$ or $\wedge^-$ and every occurrence of t with either t^+ or t^-. (Also, the

$$\begin{aligned} \text{Terms}: &\quad t, u ::= \lambda x.t \mid x\ k \mid \uparrow p \\ \text{Values}: &\quad p, q ::= x \mid \downarrow t \\ \text{Continuations}: &\quad k ::= \varepsilon \mid p :: k \mid \kappa x.t \end{aligned}$$

$$\frac{\Gamma \Uparrow \cdot \vdash t : N \Uparrow \cdot}{\Gamma \Downarrow \cdot \vdash \downarrow t : N \Downarrow \cdot}\ R_r \qquad \frac{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow t : E}{\Gamma \Uparrow \cdot \vdash t : E \Uparrow \cdot}\ S_r \qquad \frac{\Gamma \Downarrow \cdot \vdash p : P \Downarrow \cdot}{\Gamma \Uparrow \cdot \vdash \cdot \Uparrow \uparrow p : P}\ D_r \qquad \frac{}{\Gamma, x : a^+ \Downarrow \cdot \vdash x : a^+ \Downarrow \cdot}\ I_r$$

$$\frac{\Gamma, x : P \Uparrow \cdot \vdash \cdot \Uparrow t : E}{\Gamma \Downarrow P \vdash \cdot \Downarrow \kappa x.t : E}\ R_l/S_l \qquad \frac{\Gamma, x : N \Downarrow N \vdash \cdot \Downarrow k : E}{\Gamma, x : N \Uparrow \cdot \vdash \cdot \Uparrow x\ k : E}\ D_l \qquad \frac{}{\Gamma \Downarrow a^- \vdash \cdot \Downarrow \varepsilon : a^-}\ I_l$$

$$\frac{\Gamma, x : A \Uparrow \cdot \vdash t : B \Uparrow \cdot}{\Gamma \Uparrow \cdot \vdash \lambda x.t : A \supset B \Uparrow \cdot}\ \supset_r / S_l \qquad \frac{\Gamma \Downarrow \cdot \vdash p : A \Downarrow \cdot \qquad \Gamma \Downarrow B \vdash \cdot \Downarrow k : E}{\Gamma \Downarrow A \supset B \vdash \cdot \Downarrow p :: k : E}\ \supset_l$$

Figure 4 Cut-free LJF with term annotations.

polarization of propositional variables can be fixed arbitrarily.) If B is an intuitionistic theorem and $\hat{B}$ is *any* polarization of B, then there is an *LJF* proof of $\cdot \Uparrow \cdot \vdash \hat{B} \Uparrow \cdot$ [24]. Thus, polarization does not affect provability but, as we shall illustrate, it can affect the shape of proofs.

Our second extension of the proof system in Figure 1 is meant to harness the resulting variability in proofs in order to provide a rich representation for terms and formulas. Figure 4 contains the propositional *LJF* inference rules annotated with the $\lambda\kappa$-term found in [7]. This term calculus contains three syntactic categories: Terms, Values, and Continuations. Note that it is the store-left (S_l) rule that results in bindings in term structures and that such binding can result in either a λ-abstraction or a κ-abstraction.

6.2 Two normal forms for simply typed terms

If all primitive types (atomic formulas) are given a negative polarity, then the terms annotating proofs in the sequents of Figure 4 provide the usual notion of $\beta\eta$-long normal form λ-terms. Recall that terms in $\beta\eta$-*long normal form* are of the form $\lambda x_1 \ldots \lambda x_n.h\ t_1 \ldots t_m$ where h is a variable or constant, where $t_1, \ldots, t_m$ is a list of terms in $\beta\eta$-long normal form, and where the term $(h\ t_1 \ldots t_m)$ has primitive type. In particular, if we use $[\![\cdot]\!]$ to translate such λ-terms into terms of the first syntactic category in Figure 4, then

$$[\![\lambda x_1 \ldots \lambda x_n.h\ t_1 \ldots t_m]\!] = \lambda x_1 \ldots \lambda x_n.h\ (\downarrow [\![t_1]\!] :: \cdots :: \downarrow [\![t_m]\!] :: \varepsilon).$$

Note that this translation transforms the application of the function h from one argument at a time to the application of h to a list of all its arguments. Such a formal connection between $\beta\eta$-long normal forms and this style of term representation was made by Herbelin using his *LJT* sequent calculus [21]. When all primitive types are given a negative bias, then no formulas are given a positive bias and, as a result, the inference rule named R_l/S_l does not appear in such proofs and terms do not contain the κ binding operator.

▶ **Example 6.** Let i be a primitive type that will be considered negatively biased in the LJF proof system. The only terms t for which $\Gamma \Uparrow \cdot \vdash t : (i \supset i) \supset i \supset i \Uparrow \cdot$ is provable are encodings of the Church numerals. In particular, the terms corresponding to the first three numerals are $\lambda f \lambda x.x\ \varepsilon$, $\lambda f \lambda x.f\ (\downarrow (x\ \varepsilon) :: \varepsilon)$, and $\lambda f \lambda x.f\ (\downarrow (f\ (\downarrow (x\ \varepsilon) :: \varepsilon)) :: \varepsilon)$.

If all primitive types are given a positive bias, then the terms annotating proofs in the sequents in Figure 4 provide a formal definition of a normal form similar to the one described in [12] and which is commonly called *administrative normal form* (ANF).

▶ **Definition 7.** A simply typed λ-term is in administrative normal form (ANF) when written
- as $\lambda x_1 \ldots \lambda x_n.\uparrow h$, where $n \geqslant 0$ and h is a variable of primitive type
- or as $\lambda x_1 \ldots \lambda x_n.h\ (p_1 :: \cdots :: p_m :: \kappa y.t)$, where $n, m \geqslant 0$, the type of y is primitive, t is a simply typed term in ANF and values $p_1, \cdots, p_m$ are either variables of primitive type or are of the form $\downarrow t$ where t is in ANF.

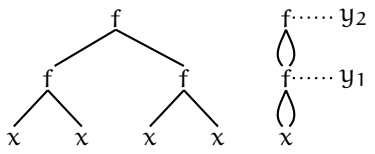
Note the following: (1) If p_i is not a variable, then it must denote a term of arrow type and, hence, it will be a λ-abstraction: that is, immediately following the $\downarrow\cdot$ there must be a λ-abstraction. (2) A closed term in ANF with a type of order 2 or less is of the form $\lambda x_1 \ldots \lambda x_n.t$ where the types of $x_1, \ldots, x_n$ are either primitive or first-order and where t does not contain any λ. It can be the case, however, that t contains κ bindings. (3) If we ignore the requirements on certain variables being of primitive type, then this definition can be extended to untyped λ-terms.

In order to facilitate the presentation of λ-terms in ANF format, we introduce the following convention. Instead of $\lambda x_1 \ldots \lambda x_n.\uparrow h$ we will simply drop the $\uparrow$ and write $\lambda x_1 \ldots \lambda x_n.h$ (remembering that h is a variable of primitive type). Also, instead of

$$\lambda x_1 \ldots \lambda x_n.h\ (p_1 :: \cdots :: p_m :: \kappa y.t) \quad \text{we write} \quad \lambda x_1 \ldots \lambda x_n.\ \mathbf{name}\ y = h\ (p_1, \ldots, p_m)\ \mathbf{in}\ t$$

(remember that y is a variable of primitive type) and where $p_1, \ldots, p_m$ is a list of either variables (of primitive types) or λ-abstractions that are also in ANF.

We use the keyword "name" here instead of "let" since let-expressions are often considered to be abbreviations for β-redexes: that is, (let $x = s$ in t) is often considered equal to $((\lambda x.t)\ s)$. Here, however, the name-expressions denote normal terms since they are annotations of cut-free sequent calculus proofs.



The figure to the right illustrates two ways of representing a labeled binary tree of height 2. Clearly, the representation on the left takes exponential space as the height increases while the representation on the right increases linearly with the height. Here we assume that x and f are two bound variables of type i and $i \to i \to i$, respectively. Choosing between these two representation schemes involves assigning either negative or positive polarity to the atomic formula (primitive type) i. For example, if i is polarized negatively, then there is an LJF proof that is annotated with the term $f\ (\downarrow(f\ (\downarrow(x\varepsilon) :: \downarrow(x\varepsilon) :: \varepsilon)) :: \downarrow(f\ (\downarrow(x\varepsilon) :: \downarrow(x\varepsilon) :: \varepsilon)) :: \varepsilon)$ which can be displayed, in a more friendly syntax, as $f\ (f\ (x,\ x),\ f\ (x,\ x))$. On the other hand, when i is polarized positively, the above term is no longer a proper annotation of an LJF proof while the term

$$\mathbf{name}\ y_1 = (f\ x\ x)\ \mathbf{in}\ \mathbf{name}\ y_2 = (f\ y_1\ y_1)\ \mathbf{in}\ y_2$$

does annotate an LJF proof. Since the ANF term format allows subterms to be shared, that format can allow for much smaller term structures. While sharing is a feature of ANF, we shall not require it to be particularly well behaved. For example, it is possible for a term in ANF to have *vacuous naming* – i.e., a named term that is never used in the name's scope – or *redundant naming* – i.e., the same term can be named more than once. For example, the term

$$\mathbf{name}\ y_1 = (f\ x\ x)\ \mathbf{in}\ \mathbf{name}\ y_2 = (f\ y_1\ y_1)\ \mathbf{in}\ \mathbf{name}\ y_3 = (f\ y_1\ y_1)\ \mathbf{in}\ y_2$$

is in ANF even though it has vacuous and redundant naming. One might imagine that multifocusing can be used to allow parallel naming, such as in the expression

$$\textbf{name } y_1 = (f\ x\ x) \textbf{ in name } y_2 = (f\ y_1\ y_1) \textbf{ and } y_3 = (f\ y_1\ y_1) \textbf{ in } y_2.$$

One might also expect that the concept of *maximal multifocusing* [8] could relate to insisting on "maximal sharing". In this paper, we shall not use multifocused proofs nor insist on the absence of vacuous or redundant naming.

6.3 Mixed term representations

The syntax of formulas of arithmetic statements depends on two primitive types: the type of formulas o and of numerals i. We present several examples of term representations below where o is polarized negatively and i is polarized positively. We also allow the binary infix term constructors $+$ and $*$ of type $i \to i \to i$ as well as the formula constructor $<$ (the less-than relation) of type $i \to i \to o$.

▶ **Example 8.** When the type i for numerals is polarized positively, the $\lambda\kappa$-calculus does not allow for expressions of the form $(s \cdots (sz) \cdots)$. Instead, encoding an expression of the form $P(2+3)$ can be done as follows:

$$\textbf{name } 1 = (s\ 0) \textbf{ in name } 2 = (s\ 1) \textbf{ in name } 3 = (s\ 2) \textbf{ in name } x = 2+3 \textbf{ in } P(x).$$

Thus, numerals are really treated as pointers into a sequence of successor terms.

▶ **Example 9.** The formula $\forall x[(x^2+6) = 5x \supset (x = 2 \vee x = 3)]$ can be written as the $\lambda\kappa$-term $\forall x[\textbf{name } y = x * x \textbf{ in name } u = 5 * x \textbf{ in name } v = y + 6 \textbf{ in } (v = u \supset (x = 2 \vee x = 3))]$.

The inversion of syntax that appears in ANF is familiar to those computing with relations instead of functions, as the following example illustrates.

▶ **Example 10.** To prove that $(2 * (5+2)) < 8+7$ in a setting with only relations (such as, say, in Prolog) one can rewrite that inequality as the following (equivalent) formulas of arithmetic.

$$\exists x(\text{plus } 5\ 2\ x \wedge \exists y(\text{times } 2\ x\ y \wedge \exists z(\text{plus } 8\ 7\ z \wedge y < z)))$$
$$\forall x(\text{plus } 5\ 2\ x \supset \forall y(\text{times } 2\ x\ y \supset \forall z(\text{plus } 8\ 7\ z \supset y < z)))$$

Here, the binary operators $+$ and $*$ are interpreted by corresponding ternary predicates.

6.4 Interpreting term constructors

As Examples 8 and 9 illustrate, arithmetic formulas can contain a mix of *uninterpreted* term constructors (for example, the constructor for numerals z and s) and *interpreted* term constructors (for example, $+$ and $*$).

The formal introduction of a new interpreted term constructor such as $f : i \to \ldots \to i \to i$ of n arguments must be tied to an interpreting μ-expression R_f of $n+1$-arity and a formal proof that R_f encodes a function, i.e.,

$$\forall \bar{x}.([\exists y.R_f(\bar{x}, y)] \wedge \forall y \forall z.[R_f(\bar{x}, y) \supset R_f(\bar{x}, z) \supset y = z]).$$

That is, achieving a proof of this theorem permits the introduction of a new constructor f where $y = f\ x_1 \ldots\ x_n$ is interpreted by $R_f\ x_1 \ldots\ x_n\ y$. In principle, this means that the

$$\frac{y, \Sigma : \Gamma \Uparrow R_f\ \bar{x}\ y, B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Gamma \Uparrow \mathbf{name}\ y = f\ \bar{x}\ \mathbf{in}\ B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{y, \Sigma : \Gamma \Uparrow R_f\ \bar{x}\ y, \Theta \vdash B \Uparrow \cdot}{\Sigma : \Gamma \Uparrow \Theta \vdash \mathbf{name}\ y = f\ \bar{x}\ \mathbf{in}\ B \Uparrow \cdot}$$

$$\frac{\Sigma : \Gamma \Uparrow \cdot \vdash \mathbf{name}\ x = f\ \bar{x}\ \mathbf{in}\ B \Uparrow \cdot}{\Sigma : \Gamma \Downarrow \cdot \vdash \mathbf{name}\ x = f\ \bar{x}\ \mathbf{in}\ B \Downarrow \cdot} \qquad \frac{\Sigma : \Gamma \Uparrow \mathbf{name}\ x = t\ \mathbf{in}\ B \vdash \cdot \Uparrow \Delta}{\Sigma : \Gamma \Downarrow \mathbf{name}\ x = t\ \mathbf{in}\ B \vdash \cdot \Downarrow \Delta}$$

Figure 5 Introduction rules for the constructor f and the relation R_f which interprets it.

Name binding rules: the variable x is not bound in Σ nor in Ψ.

$$\frac{\Sigma : x := t, \Psi; \Gamma \Uparrow B, \Theta \vdash \Delta_1 \Uparrow \Delta_2}{\Sigma : \Psi; \Gamma \Uparrow \mathbf{name}\ x = t\ \mathbf{in}\ B, \Theta \vdash \Delta_1 \Uparrow \Delta_2} \qquad \frac{\Sigma : x := t, \Psi; \Gamma \Uparrow \cdot \vdash B \Uparrow \cdot}{\Sigma : \Psi; \Gamma \Uparrow \cdot \vdash \mathbf{name}\ x = t\ \mathbf{in}\ B \Uparrow \cdot}$$

$$\frac{\Sigma : x := t, \Psi; \Gamma \Downarrow \cdot \vdash B \Downarrow \cdot}{\Sigma : \Psi; \Gamma \Downarrow \cdot \vdash \mathbf{name}\ x = t\ \mathbf{in}\ B \Downarrow \cdot} \qquad \frac{\Sigma : x := t, \Psi; \Gamma \Downarrow B \vdash \cdot \Downarrow E}{\Sigma : \Psi; \Gamma \Downarrow \mathbf{name}\ x = t\ \mathbf{in}\ B \vdash \cdot \Downarrow E}$$

Positive phase quantifier rules

$$\frac{\Sigma, \Sigma(\Psi) \Uparrow \cdot \vdash t : \tau \Uparrow \cdot \quad \Sigma : \Psi; \Gamma \Downarrow [t/x]B \vdash \cdot \Downarrow E}{\Sigma : \Psi; \Gamma \Downarrow \forall x_\tau.B \vdash \cdot \Downarrow E} \qquad \frac{\Sigma, \Sigma(\Psi) \Uparrow \cdot \vdash t : \tau \Uparrow \cdot \quad \Sigma : \Psi; \Gamma \Downarrow \cdot \vdash [t/x]B \Downarrow \cdot}{\Sigma : \Psi; \Gamma \Downarrow \cdot \vdash \exists x_\tau.B \Downarrow \cdot}$$

Figure 6 The incorporation of the *naming* context Ψ.

formula ($\mathbf{name}\ y = f\ x_1 \ldots x_n\ \mathbf{in}\ B$) is interpreted as either $\forall y.(R_f\ x_1 \ldots x_n\ y \supset B)$ or $\exists y.(R_f\ x_1 \ldots x_n\ y \wedge^+ B)$. Clearly, the naming construction is a *self-dual* operator on formulas in the sense that $\neg(\mathbf{name}\ y = f\ x_1 \ldots x_n\ \mathbf{in}\ B)$ is equivalent to $(\mathbf{name}\ y = f\ x_1 \ldots x_n\ \mathbf{in}\ \neg B)$. As a result, such formulas are said to have an *ambiguous* polarity since they can be coerced to be negative or positive. The introduction rules for interpreted term constructors are given in Figure 5.

6.5 A final extension

In order to treat the naming (sharing) of structures built using uninterpreted symbols within proofs and computations, we need to add to our sequents (both $\Uparrow$ and $\Downarrow$) an additional zone (using the Ψ syntactic variable) that explicitly retains the naming relation. We do this by adding the Ψ context to all the previous arithmetic-related sequents and inference rules. We also add the inference rules that appear in Figure 6. In the first four of these inference rules, the formula-level binder $\mathbf{name}\ y = t\ \mathbf{in}\ B$ is translated to a proof-level binder by adding the pair $y := t$ to the Ψ context.

The quantifier rules that instantiate their quantifier with a term are modified in Figure 6 so that the naming structure of sequents is respected. In particular, those rules employ the premise $\Sigma, \Sigma(\Psi) \Uparrow \cdot \vdash t : \tau \Uparrow \cdot$. (Here, $\Sigma(\Psi)$ is the set of (typed) variables that are bound in Ψ.) Thus, the term t is, in general, a $\lambda\kappa$-term. The inference rules for equality must also be changed in order to account for the treatment of $\lambda\kappa$-terms: with only first-order constructors present (such as in our treatment of natural numbers), the treatment of unification in this setting can be based on the Martelli-Montanari algorithm [25].

7 Conclusion

We have presented a treatment of functional computation based on relations. Principles in proof theory provided both a method for moving expressions denoting embedded computation into naming-combinators of the logic (ANF normal form) and a means of organizing Gentzen-style introduction rules so that functional computations can be identified as one specific phase of computation (the negative phase). Since this view of computation is based on the construction of cut-free proofs, it is rather different from, say, the Curry-Howard correspondence.

While we have illustrated most of this mechanism using first-order term structures (such as Peano's numerals), the proof theory behind *LJF* works at all finite types. As a result, this approach to functional computation is a possible avenue to explore how functional programming might be extended to treat terms containing λ-bindings.

The proof theory presented here is compatible with the proof theory for least and greatest fixed points that has been developed in a series of papers [26, 14, 15, 30] and in the Abella theorem prover [6, 1, 13]. A possible practical consequence of the design in this paper is an avenue for adding to Abella functional computations via the addition of interpreted term constructors.

Acknowledgments. We thank Beniamino Accattoli, Roberto Blanco, and the anonymous reviewers for their comments on an earlier draft of this paper.

References

1 The Abella prover, 2012. Available at `http://abella-prover.org/`.

2 Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992. `doi:10.1093/logcom/2.3.297`.

3 Ali Assaf, Guillaume Burel, Raphaël Cauderlier, David Delahaye, Gilles Dowek, Catherine Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. Dedukti: a logical framework based on the λΠ-calculus modulo theory. Unpublished, 2016. URL: `http://www.lsv.ens-cachan.fr/~dowek/Publi/expressing.pdf`.

4 David Baelde. *A linear approach to the proof-theory of least and greatest fixed points*. PhD thesis, Ecole Polytechnique, December 2008. URL: `http://www.lix.polytechnique.fr/~dbaelde/thesis/`.

5 David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. on Computational Logic*, 13(1), April 2012. `doi:10.1145/2071368.2071370`.

6 David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. Abella: A system for reasoning about relational specifications. *Journal of Formalized Reasoning*, 7(2), 2014. `doi:10.6092/issn.1972-5787/4650`.

7 Taus Brock-Nannestad, Nicolas Guenot, and Daniel Gustafsson. Computation in focused intuitionistic logic. In Moreno Falaschi and Elvira Albert, editors, *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming, Siena, Italy, July 14–16, 2015*, pages 43–54, 2015. `doi:10.1145/2790449.2790528`.

8 Kaustuv Chaudhuri, Dale Miller, and Alexis Saurin. Canonical sequent proofs via multi-focusing. In G. Ausiello, J. Karhumäki, G. Mauri, and L. Ong, editors, *Fifth International Conference on Theoretical Computer Science*, volume 273 of *IFIP*, pages 383–396. Springer, September 2008.

9 Alonzo Church. A formulation of the Simple Theory of Types. *J. of Symbolic Logic*, 5:56–68, 1940.

10 Denis Cousineau and Gilles Dowek. Embedding pure type systems in the lambda-Pi-calculus modulo. In Simona Ronchi Della Rocca, editor, *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4583 of *LNCS*, pages 102–117. Springer, 2007.

11 Roy Dyckhoff and Stephane Lengrand. Call-by-value λ-calculus and LJQ. *J. of Logic and Computation*, 17(6):1109–1134, 2007.

12 Cormac Flanagan, Amr Sabry, Bruce F. Duba, and Matthias Felleisen. The essence of compiling with continuations. *ACM SIGPLAN Notices*, 28(6):237–247, 1993. URL: citeseer.nj.nec.com/174731.html.

13 Andrew Gacek. The Abella interactive theorem prover (system description). In A. Armando, P. Baumgartner, and G. Dowek, editors, *Fourth International Joint Conference on Automated Reasoning*, volume 5195 of *LNCS*, pages 154–161. Springer, 2008. URL: http://arxiv.org/abs/0803.2305.

14 Andrew Gacek, Dale Miller, and Gopalan Nadathur. Combining generic judgments with recursive definitions. In F. Pfenning, editor, *23th Symp. on Logic in Computer Science*, pages 33–44. IEEE Computer Society Press, 2008. URL: http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lics08a.pdf.

15 Andrew Gacek, Dale Miller, and Gopalan Nadathur. Nominal abstraction. *Information and Computation*, 209(1):48–73, 2011. doi:10.1016/j.ic.2010.09.004.

16 Gerhard Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland, Amsterdam, 1935. doi:10.1007/BF01201353.

17 Jean-Yves Girard. A new constructive logic: classical logic. *Math. Structures in Comp. Science*, 1:255–296, 1991. doi:10.1017/S0960129500001328.

18 Jean-Yves Girard. A fixpoint theorem in linear logic. An email posting to the mailing list linear@cs.stanford.edu, February 1992.

19 Michael J. Gordon, Arthur J. Milner, and Christopher P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, volume 78 of *LNCS*. Springer, 1979.

20 Quentin Heath and Dale Miller. A proof theory for model checking: An extended abstract. In Iliano Cervesato and Maribel Fernández, editors, *Proceedings Fourth International Workshop on Linearity (LINEARITY 2016)*, volume 238 of *EPTCS*, January 2017. doi:10.4204/EPTCS.238.1.

21 Hugo Herbelin. A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Computer Science Logic, 8th International Workshop, CSL'94*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 1995.

22 Hugo Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de lambda-termes et comme calcul de stratégies gagnantes*. PhD thesis, Université Paris 7, 1995.

23 D. Hilbert and P. Bernays. *Grundlagen der Mathematik II*. Springer Verlag, 1939.

24 Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009. doi:10.1016/j.tcs.2009.07.041.

25 Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982.

26 Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:91–119, 2000. doi:10.1016/S0304-3975(99)00171-1.

27 Dale Miller and Alexis Saurin. A game semantics for proof search: Preliminary results. In *Proceedings of the Mathematical Foundations of Programming Semantics (MFPS05)*, number 155 in ENTCS, pages 543–563, 2006.

28 Peter Schroeder-Heister. Rules of definitional reflection. In M. Vardi, editor, *8th Symp. on Logic in Computer Science*, pages 222–232. IEEE Computer Society Press, IEEE, June 1993. doi:10.1109/LICS.1993.287585.

29 Robert J. Simmons and Frank Pfenning. Weak focusing for ordered linear logic. Technical Report CMU-CS-10-147, Carnegie Mellon University, April 2011.

30 Alwen Tiu and Alberto Momigliano. Cut elimination for a logic with induction and co-induction. *Journal of Applied Logic*, 10(4):330–367, 2012. doi:10.1016/j.jal.2012.07.007.

31 Alwen Tiu, Gopalan Nadathur, and Dale Miller. Mixing finite success and finite failure in an automated prover. In *Empirically Successful Automated Reasoning in Higher-Order Logics (ESHOL'05)*, pages 79–98, December 2005.

32 Anne Sjerp Troelstra, editor. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer, 1973.

33 Alexandre Viel and Dale Miller. Proof search when equality is a logical connective. Presented to the International Workshop on Proof-Search in Type Theories, July 2010. URL: http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/unif-equality.pdf.

Diagrammatic Semantics for Digital Circuits*

Dan R. Ghica[1], Achim Jung[2], and Aliaume Lopez[3]

1 **University of Birmingham, United Kingdom**
2 **University of Birmingham, United Kingdom**
3 **ENS Cachan, Université Paris-Saclay, France**

Abstract

We introduce a general diagrammatic theory of digital circuits, based on connections between monoidal categories and graph rewriting. The main achievement of the paper is conceptual, filling a foundational gap in reasoning syntactically and symbolically about a large class of digital circuits (discrete values, discrete delays, feedback). This complements the dominant approach to circuit modelling, which relies on simulation. The main advantage of our symbolic approach is the enabling of automated reasoning about *parametrised circuits*, with a potentially interesting new application to *partial evaluation* of digital circuits. Relative to the recent interest and activity in categorical and diagrammatic methods, our work makes several new contributions. The most important is establishing that categories of digital circuits are Cartesian and admit, in the presence of feedback expressive iteration axioms. The second is producing a general yet simple graph-rewrite framework for reasoning about such categories in which the rewrite rules are computationally efficient, opening the way for practical applications.

1998 ACM Subject Classification B.6.3 Hardware description languages

Keywords and phrases digital circuits, string diagrams, rewriting, operational semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.24

1 Introduction

Of the many differences between the worlds of software and hardware design, a particularly intriguing one is their prevailing modelling methodologies. The workhorse of software reasoning – *operational semantics* [28] – is syntactic and reduction-based. It is essentially an abstract, entirely machine-independent presentation of a programming language which is not required to be faithful to the execution model other than insofar as the final result is concerned. On the other hand, reasoning about hardware relies on having an accurate execution model, akin to what we would call an *abstract machine* in programming languages, usually some kind of automaton [21]. To reason about a circuit, it is translated so that its execution is simulated by the automaton. The abstract machine approach is of course established and useful in programming language theory as well [23], especially in compiler design. But the operational semantics has several advantages over the abstract machine approach, of which perhaps the most important is the ability to evaluate programs which are specified only in part. This is useful because many front-end compiler optimisations are, in one way or another, partial evaluations [7].

Broadly speaking, the main contribution of our paper is to provide an operational semantics for digital circuits, based on diagram rewriting. Our methodology is influenced by the interplay between graph rewriting and monoidal categories, which led in the last

* This work has been supported in part by EPSRC grant EP/P004490/1.

decade to diagrammatic models for quantum computing [1], signal flow [4] and asynchronous circuits [11]. Algebraic specifications in the style of monoidal categories have been pioneered by Sheeran in the 1980s [30] and a certain amount of algebraic reasoning about circuits using such specifications has been attempted [25]. However, a full and systematic categorical presentation of digital circuits has only been given recently by the first two authors of the present paper [13]. Starting only from an equational specification of digital components ("gates") it shows that the free traced monoidal category, subject to certain quotients, is Cartesian. Such categories, known as *dataflow categories* [29, Sec. 6.4] (or *traced cartesian categories* [15]), have very useful equations for iterative unfolding of the trace [6, 15, 31], offering a convenient way to model feedback.

The main theoretical contribution of this paper is providing a rewriting semantics for dataflow categories with a discrete delay operator. It is well known that an algebraic semantics does not automatically translate into an operational semantics, because distributive laws (in particular the functoriality of the tensor product) are directionless. This is where the diagrammatic approach can help when used as a graphical syntax, by avoiding the need for such problematic laws [3] and leading to computationally efficient rewriting. The iteration axioms also raise difficulties, this time of identifying diagrammatic redexes. This problem is compounded by the fact that choosing the wrong iterators to unfold can lead to unproductive rewrites. Finally, the presence of delays raises yet a different set of technical challenges because they cannot be rewritten out of a circuit but only moved around using a *retiming* axiom [24]. We solve these problems by writing circuit diagrams in a particular canonical form, which we call *global trace delay*, for which we can provide effective and efficient unfolding, with certain guarantees of productivity.

The main motivation of this work is to open the door to new optimisation techniques for digital circuits, similar to partial evaluation. We will test our theory against a particularly challenging class of circuits, so called *circuits with combinational feedback* [26]. These are circuits which, despite the presence of feedback loops, behave just like combinational circuits, i.e. they exhibit none of the effects associated with genuine feedback, such as state or oscillation. As is the case with operational semantics, we will see how handling such circuits is mathematically elementary and fully automated. This is indeed remarkable, because the conventional automata-based reasoning method does not accept combinational feedback. Denotational semantics can model such circuits [27] but using rather complex mathematical machinery. Moreover, we will show how circuits with combinational feedback which are parametrised by unspecified "black box" components can be just as easily handled by our approach. As far as we know, there is no existing method for modelling such circuits (called "abstract circuits") in the design literature.

Note. A longer version of this including all proofs is available as a technical report [12].

2 Categorical semantics background

A categorical semantics of circuits was given by the first two authors in an earlier paper [13]. In this section we cover the essential results required to justify the diagrammatic semantics.

2.1 Combinational circuits

Let *object variables*, labelling (collections of) wires, be natural numbers and let *morphism variables* be labels for boxes (e.g., gates and circuits). This is a category of PROducts and Permutations (PROP) [22].

▶ **Definition 1.** Let **Circ** be a categorical signature with objects the natural numbers $\mathbb{N}$ and a finite set of morphisms which may be grouped into the following three classes:

- *levels* (or *values*) $v : 0 \to 1$ forming a lattice $(\mathbf{V}, \sqsubseteq)$;
- *gates* $k : m \to 1$; and
- the special morphisms *join* $\curlyvee : 2 \to 1$, *fork* $\curlywedge : 1 \to 2$, and *stub* $\mathsf{w} : 1 \to 0$.

All circuit signatures include combinators for joining two outputs (*join*) and duplicating an input (*fork*), as well as the ability to discard an output (*stub*). What varies from signature to signature is the number of signal levels and the set of gates. Since levels form a lattice, they must include a smallest element ($\bot$), corresponding to a disconnected input, and a top element ($\top$) corresponding to an illegal output ("short circuit"). In the simplest and most common instance, the set of level has two other elements, *high* and *low*, but it can go beyond that. For example, in the case of metal-oxide-semiconductor field-effect transistors (MOSFET) it makes sense, in certain designs, to model the diode properties of the transistor by taking into account four levels (strong and weak high and low voltage, cf. the relevant IEEE standard for logical simulations [17]).

Circuits, in the sense of this paper, are the morphisms of a free categorical construction over their signature. Beginning with *combinational circuits*, the free construction is as follows:

▶ **Definition 2.** Let **CCirc** be the free symmetric monoidal category over **Circ** and monoidal signature $(\mathbb{N}, +, 0)$, and equations:

Fork: $\curlywedge \circ v = v \otimes v$.

Join: $\curlyvee \circ (v \otimes v') = v \sqcup v'$.

Stub: $\mathsf{w} \circ v = id_0$.

Gate: $k \circ \bigotimes_{i=1,m} v_i = v$, such that whenever $v_i \sqsupseteq v'_i$ then $k \circ \bigotimes_{i=1,m} v_i \sqsupseteq k \circ \bigotimes_{i=1,m} v'_i$.

We will call morphisms in this category *combinational circuits.*

The first three equations model the fact that a fork duplicates a value, a join coalesces two values, and a stub discards anything it receives. The gate equations must cover all possible inputs to a gate k and their particular format entails that the output from a gate is always one of the original levels in $\mathbf{V}$. Since $\mathbf{V}$ is a lattice, the monotonicity requirement is also expressible equationally.

It is known that, in a formal sense, the equality of morphisms in a free SMC corresponds to graph isomorphisms in the diagrammatic language [18], where diagrams are created by the operations of sequential composition ($\circ$), parallel composition ($\otimes$) and symmetry ($\mathsf{x}_{m,n}$, the swapping of two buses with m and n wires, respectively), governed by coherence equations. We will usually write composition in diagrammatical order $f \cdot g = g \circ f$. We write the identity (bus of width m) $\mathrm{id}_m : m \to m$ as simply m. For simplicity we also write $\bigotimes_{i=1,m} f = f^m$, $\bigotimes_{i=1,m} f_i = \mathbf{f}$ and $\bigotimes_{i=1,m} v_i = \mathbf{v}$. For lack of space we will not enumerate the coherence equations here, since they are standard.

The *Gate* axioms state that the behaviour of basic components is fully defined by their inputs, i.e. they are *extensionally complete*. By simple inductive arguments on the structure of morphisms we can establish that all circuits are in fact extensionally complete, i.e. for any circuit (not just gates) $f : m \to n$, for any values $v_i, 1 \leq i \leq m$, there exist unique values $v'_j, 1 \leq j \leq n$ such that $f \circ \bigotimes_{i=1,m} v_i = \bigotimes_{i=1,n} v'_i$. Intuitively this means that we only model *local interactions*, abstracting away from global effects such as electromagnetic interference or quantum tunnelling etc.

We can further say that two circuits with the same input-output behaviour are *extensionally equivalent*, and a simple inductive argument shows that this is a *congruence*, i.e. it is an equivalence preserved by sequential and parallel composition. Therefore it makes sense to

quotient our category **CCirc** and create a new category **ECCirc** in which equivalent circuits are made equal.

ECCirc has interesting additional categorical properties which aid reasoning. Two are of particular importance. The first one is that **ECCirc** is *Cartesian*. The *diagonals* are defined by $\Delta_0 = 0$ and $\Delta_{n+1} = (\Delta_n \otimes \curlywedge) \cdot (n \otimes \mathsf{x}_{(1,n)} \otimes 1)$, *forks* of width n. The diagonal must satisfy two *coherences*: $\langle f, f \rangle = \Delta_n \cdot (f \otimes f) = f \cdot \Delta_m$and $f \cdot \mathsf{w}^m = \mathsf{w}^m$.

Another useful property is that $(\curlywedge, \curlyvee, \mathsf{w}, \bot)$ forms what is known as a *bialgebra*, i.e. an algebraic structure in which $(\curlyvee, \bot)$ is a commutative monoid, $(\curlywedge, \mathsf{w})$ is a co-commutative co-monoid, such that $\curlyvee \cdot \curlywedge = \curlywedge^2 \cdot (1 \otimes \mathsf{x}_{1,1} \otimes 1) \cdot \curlyvee^2$. Finally, fork is a section and join a retraction, $\curlywedge \cdot \curlyvee = 1$, but are not generally isomorphisms. A convenient derived connector is the *join* of width n, defined as $\nabla_0 = 0$ and $\nabla_{n+1} = (n \otimes \mathsf{x}_{1,n} \otimes 1) \cdot (\nabla_n \otimes \curlyvee)$.

2.2 Circuits with discrete delays

▶ **Definition 3.** Let $\mathbf{CCirc}_\delta$ be the category obtained by freely extending **ECCirc** with a new morphism $\delta : 1 \to 1$ subject to the following equations:
Timelessness: For any gate or structural morphism $k : m \to n$, $\delta^m \cdot k = k \cdot \delta^n$.
Streaming: For any gate $k : m \to 1$ and levels $\mathbf{v}$, $(\delta^m \otimes \mathbf{v}) \cdot \nabla_m \cdot k = ((\delta^m \cdot k) \otimes (\mathbf{v} \cdot k)) \cdot \nabla_1$.
Disconnect: $\bot \cdot \delta = \bot$.
Unobservable delay: $\delta \cdot \mathsf{w} = \mathsf{w}$.
Timelessness means that compared to δ, all other basic gates and structural morphisms compute instantaneously. An immediate consequence is that delays can be propagated through combinational circuits, akin to *retiming* [24]. *Disconnect* means that the initial conditions of circuits is $\bot$, so that a wire that also promises to dangle later might as well be considered dangling already. The last rule expresses the same for dangling output wires.

The *Streaming* axiom is more interesting, and it was one of the essential new axioms proposed in [13]. It is key to capturing the intuition of δ as a *delay* operator. Mathematically, first observe that there are infinitely many morphisms of type $0 \to 1$ in $\mathbf{CCirc}_\delta$, not just the finitely many values. This is because expressions such as $v \cdot \delta$ do not reduce to a value. However, it can be shown that any expression built from values, δ, and the structural morphisms can be transformed into a *canonical form* which may be viewed as a *sequence of values presented over time*, something that is called a *waveform* in hardware design lingo. We write a waveform consisting of $n+1$ values as a list $s_n = v_n :: v_{n-1} :: \cdots :: v_0$ where v_n is the value that is currently visible, v_{n-1} becomes visible in the next step, and so on. Formally, $s_0 = v_0$ and $s_{n+1} = (s_n \cdot \delta \otimes v_{n+1}) \cdot \curlyvee$. For example, the expression $v \cdot \delta$ corresponds to the waveform $\bot :: v$; a value v is equal to (any of) the waveforms $v :: \bot :: \cdots :: \bot$ which means that it is only available *now* but no longer in the next time-step. As before, we write $\bigotimes_{i=i,m} s = s^m$ and $\bigotimes_{i=i,m} s_i = \mathbf{s}$.

The *Streaming* axiom now tells us how a gate processes a waveform: we create two separate instances of the gate, one to process the immediate inputs and another to process the subsequent inputs. Applying it repeatedly to a given circuit allows us to determine the waveform that is produced at the output wires. We obtain:

▶ **Theorem 4** (Extensionality)**.** *Given any morphism f in $\mathbf{CCirc}_\delta$, for any input waveform $\mathbf{s}$ there exists a unique output waveform $\mathbf{s}'$ such that $\mathbf{s} \cdot f = \mathbf{s}'$.*

As in the case of circuits without delays, we can show that extensionality is a congruence and we can quotient by it, creating an *extensional* category of circuits with delays, $\mathbf{ECCirc}_\delta$. It is then a routine exercise to show $\mathbf{ECCirc}_\delta$ is Cartesian, with the diagonal and terminal object defined the same way as in **ECCirc**.

2.3 Circuits with feedback

▶ **Definition 5.** Let $\mathbf{CCirc}^*_\delta$ be the category obtained from $\mathbf{ECCirc}_\delta$ by freely adding a trace operator.

Diagrammatically, the trace operator applied to a diagram $f : m + k \to n + k$ corresponds to a feedback loop of width k, written $\mathrm{Tr}^k(f) : m \to n$. Symmetric traced monoidal categories (STMC) satisfy a number of equations (coherences) which we will not enumerate for lack of space [19]. As before, their interpretation coincides with equality of diagrams (with feedbacks) up to graph isomorphism.

As before, we are committed to an extensional view of circuits where the only observable is the input-output behaviour. In combinational circuits, with or without delays, the only way we can create a circuit with 0 outputs is by explicitly composing a circuit $f : m \to n$ with w^n. However, 0-output circuits can arise in more complicated ways in the presence of feedback, whenever all the outputs are fed back. It is convenient and reasonable to equate all 0-output circuits to w^n, trivially a congruence. The new quotient category is called $\mathbf{OCirc}^*_\delta$. In this category all diagrams of shape $f : m \to 0$ are therefore equal which, categorically speaking, makes 0 a "*terminal object*".

In general, in programs feedback corresponds to recursion and iteration, and syntactic models (operational semantics) of such programs involve creating two copies of the code recursed over. For example, the operational semantics of the Y-combinator as applied to some G is $YG = G(YG)$. A similar rule does not exist in general for SMTCs unless the category is also Cartesian. Such categories, also called *data-flow categories* [6], admit an *iterator* defined for any $f : m + n \to n$, $\mathrm{iter}^n(f) = \mathrm{Tr}^n(f \cdot (\Delta_n \otimes n)) : m \to n$, which satisfies
Naturality : $\mathrm{iter}((g \otimes n) \cdot f) = g \cdot \mathrm{iter}(f)$ for any $g : k \to m$,
Iteration : $\mathrm{iter}(f) = \langle m, \mathrm{iter}(f)\rangle \cdot f$
Diagonal : $\mathrm{iter}^n(\mathrm{iter}^n(f)) = \mathrm{iter}^n((\langle n, n\rangle \otimes m) \cdot f)$.

We can use these equations because the category in which we operate is indeed Cartesian.

▶ **Theorem 6** ([13]). *The category* $\mathbf{OCirc}^*_\delta$ *is Cartesian with diagonal* Δ_n.

To conclude the section, we discuss the existence of a concrete model for $\mathbf{OCirc}^*_\delta$ which will confirm the axiomatic framework is consistent. It needs to be Cartesian and support the delay operator and iteration. The usual example of a traced SMC, sets and relations, is not Cartesian so a slightly more complex construction is required. We start with a basic model for combinational circuits based on the lattice $\mathbf{V}$ of values (Def. 1).

▶ **Definition 7.** Let $\mathcal{V}$ be the category whose objects are finite powers of $\mathbf{V}$ and whose morphisms are monotone maps.

▶ **Theorem 8** ([12]). *There is a unique traced monoidal functor* $[\![\cdot]\!]^{\mathcal{S}}$ *from* $\mathbf{CCirc}^*_\delta$ *to* $\mathcal{S}$ *mapping the object* 1 *of the former to* $\mathbf{V}$ *of the latter.*

3 Diagrammatic operational semantics

The results of the previous section establish a powerful framework for algebraic reasoning about circuits. However, this framework is not equally useful for *automatic* reasoning and cannot implement a reasonable operational semantics.

The first obstacle is the functoriality property of the tensor, which lacks directionality. Consider the circuit corresponding to the boolean expression $t \wedge f \wedge t$, where the constants involved satisfy the obvious equations. This diagram can be specified in several ways. Some of

the specifications, e.g. $(((t \otimes f) \cdot \wedge) \otimes t) \cdot \wedge$ have the immediately identifiable redex $(t \otimes f) \cdot \wedge = f$ which reduces the overall expression to $(f \otimes f) \cdot \wedge$, which reduces to f. However, the same circuit can be equivalently written as $(t \otimes f \otimes t) \cdot (\wedge \otimes id) \cdot \wedge$ which has no obvious redex. Finding redexes in such structural diagram specifications is computationally prohibitive and an unsuitable operational semantics. The alternative is to exploit the connection between monoidal categories in general, and traced monoidal categories in particular, and certain graphs. This idea has been analysed in depth recently [3].

We will give a concrete presentation of the graphs following Kissinger's *framed point graphs*, which are a free (strict) symmetric traced monoidal category [20, Thm. 5.5.10]. To make the presentation more accessible we will elide some of the categorical technicalities in *loc. cit.* and give a more direct presentation.

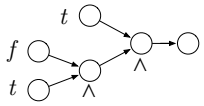
Let a labelled directed acyclic graph (LDAG) be a DAG (V, E) equipped with a partial labelling function $f : V \rightharpoonup L$. Let a labelled interfaced DAG (*LIDAG*) be a labelled DAG with two distinguished lists of unlabelled nodes representing the "input" (I) and "output" (O) interfaces, $G = (V, E, f, I, O)$. We write the set of elements of a list L as $|L|$, and list concatenation and cons both as $- :: -$. Let the *zip* operation on lists be defined as usual, $zip\ nil\ nil = nil$, and $zip\ x :: xs\ y :: ys = (x, y) :: zip\ xs\ ys$.

Unlabelled nodes are called *wire nodes* and edges connecting them are called *wires*. A *wire homeomorphism* [20, Sec. 5.2.1] is any insertion or removal of wire nodes along wires, which does not change the shape of the graph. Two *LIDAG*s are considered to be equivalent if they are graph isomorphic up to renaming vertices and wire homeomorphisms:

$$(V \uplus \{a, b, c\}, E \uplus \{(a, b), (b, c)\}, f, I, O) \simeq (V \uplus \{a, c\}, E \uplus \{(a, c)\}, f, I, O),$$

if and only if $f(b)$ is undefined. The quotienting of *LIDAG*s by this equivalence gives us *framed point graphs* (FPG) [20, Def. 5.3.1].

The algebraic specifications of the diagrams associated with the expression $t \wedge f \wedge t$ mentioned above all correspond to the (same) framed point graph with empty input interface and 1-point output interface:

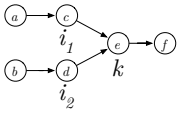


This representation solves the problem raised by the functoriality of the tensor, as redexes can be detected in linear time in the size of the graph. Note that all wire nodes can also be removed in linear time in terms of the size of the graph.

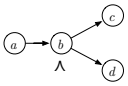
Sequential composition of two FPGs where the size of the output of the first matches the size of the input of the second is defined by identifying the output list and the input list of the two graphs. Since FPGs are equal up to renaming of vertices, the names of the wires can be chosen so that the composition is well defined. The unlabelled input and output nodes become wire nodes in the composition. The tensor is the disjoint union of the two graphs. It is always well defined since graphs are identified up to vertex renaming. The trace operator picks the head nodes of the input and output lists of points, makes them wire nodes, and connects them. Formally, if $G_i = (V_i, E_i, f_i, I_i, O_i)$ and $G = (V, E, f, a :: I, b :: O)$ then

$$\begin{aligned} G_1; G_2 &= (V_1 \uplus V_2, E_1 \uplus E_2 \uplus |zip\ O_2\ I_1|, f_1 \uplus f_2, I_1, O_2) \\ G_1 \otimes G_2 &= (V_1 \uplus V_2, E_1 \uplus E_2, f_1 \uplus f_2, I_1 :: I_2, O_1 :: O_2) \\ \mathrm{Tr}(G) &= (V, E \uplus \{(b, a)\}, f, I, O). \end{aligned}$$

Constants are interpreted by the graphs below:



$k : 2 \to 1$



$\curlywedge : 1 \to 2$

A binary gate is shown, but n-ary gates and *join* are similar. The labels i_1, i_2 are required to identify inputs on non-commutative constants but can be otherwise omitted (e.g. in the case of join $\curlyvee : 2 \to 1$). If unambiguous we shall not display the node identities $(a, b, \ldots)$ just their labels, if any.

The graph representation provides a solution for dealing with the functoriality of the tensor, but the presence of feedback raises a new, additional problem. Suppose that we deal with a graph which includes several iterations, e.g. $\mathrm{iter}(f) \cdot \mathrm{iter}(g)$. This graph raises two computationally difficult questions. The first one is how we identify feedback patterns efficiently so that we can apply the iteration axiom. The second one is, if there are several instances of the iteration unfolding axiom that can be applied, what is the schedule of applying them? Without a good (linear time) solution to the first problem we cannot claim that we have a genuine operational semantics. Without a good solution to the second problem we run into technical problems of termination and confluence. Diagrammatic representation alone is no longer the solution.

The main contribution of this section, and of the paper, is showing how to solve these two problems.

Before we proceed, we will give a generalisation of the Streaming axiom which will aid the formulation of the diagrammatic semantics, which relies in turn on a general property which holds in all free symmetric monoidal categories which we call *staging*.

▶ **Lemma 9** (Staging). *Given a free SMC over a signature Γ, any morphism f can be written as a sequence of compositions $f = f_0 \cdot f_1 \cdots f_n$ where f_i is a tensor including exactly one non-identity morphism, $f_i = m \otimes k \otimes n$.*

Let us call *passive* a circuit which has no occurrences of a value. Using the Staging Lemma (9) we can show that:

▶ **Lemma 10** (Generalised Streaming). *For any passive combinational circuit $f : m \to n$, $(\delta^m \otimes m) \cdot \nabla_m \cdot f = (f \cdot \delta^n \otimes f) \cdot \nabla_n$.*

Moreover, a diagram with feedback loops can always be rewritten as single, global, feedback loop.

▶ **Lemma 11** (Global trace form). *For any morphism f in a free STMC there exists a trace-free morphism $\hat{f}$ such that $Tr^n(\hat{f}) = f$ for some $n \in \mathbb{N}$.*

This form can be maintained in the graph representation with constant overhead. In the graph we can maintain a distinguished subset of known *feedback* wire nodes so that the feedback loops can be immediately identified. This can be done compositionally just by keeping track of the feedback wire nodes in sequential composition, tensor and trace. By maintaining the feedback wire nodes explicitly we can ensure two useful invariants. First, the rest of the graph is a DAG. Second, for each feedback wire node there is precisely one incoming and one outgoing edge. We call these graphs *trace-framed point graphs* (TFPGs). Note that feedback wire nodes must not be entirely removed as wire homeomorphisms are applied. Feedback edges that bypass the set of feedback wire nodes are legal, but break the TFPG form. Maintaining these restrictions is computationally trivial (constant overhead).

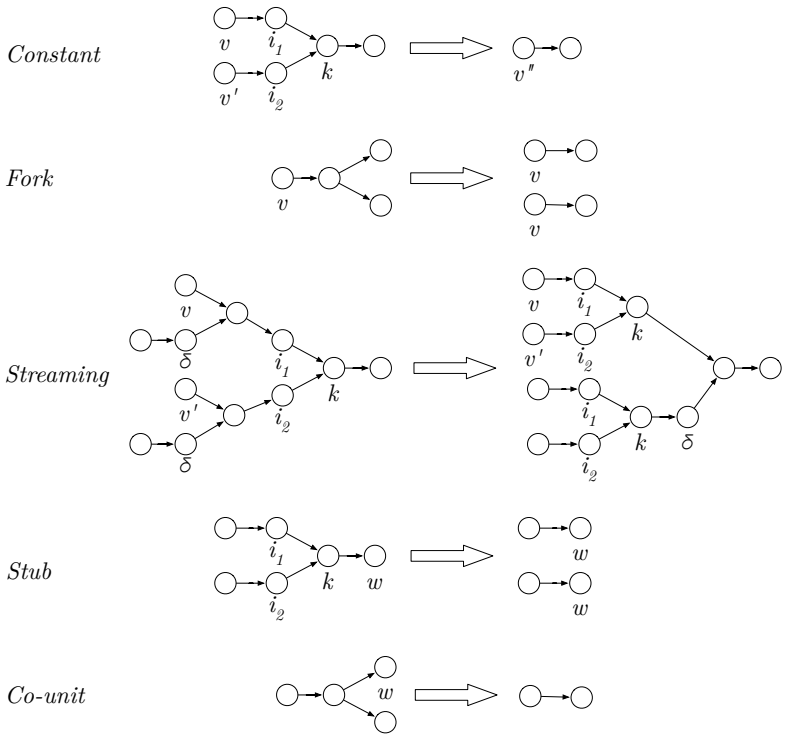


Figure 1 Rewrite rules for combinational circuits.

We are now in a position to define the diagrammatic semantics as a graph-rewriting system in which each rule can be applied efficiently, in linear time as a function of the size of the graph.

3.1 Rewrite rules for combinational circuits

The categorical equations can be expressed as graph rewrite rules, summarised in Fig. 1. We give the rules in an informal diagrammatic style, but a formalisation in an established formalism such as DPO [8] is a standard exercise.

The *Constant* rule shown is for binary constants, but rules for constants of different arity are similar. In the case of the *Constant* rule we require $v'' = (v \otimes v') \cdot k$.

Enhanced Constant Rules. Besides the basic equations for constants, more equations can be proved by extensionality in which reductions can be carried out without all input values being present. For example, $true \vee x = true$ or $true \wedge x = x$. These equations are admissible in the rewrite system.

We call the rewrite rules above the *local rewrite rules*. A TFPG where no local rewrite rules apply is in *canonical form*. The following basic properties of the rewrite system hold.

▶ **Proposition 12.** *The local rewrite rules are sound relative to the categorical equations.*

▶ **Lemma 13.** *The local rewrite rules are strongly confluent.*

▶ **Lemma 14** (Progress). *A circuit $f : 0 \to n, n \neq 0$ without traces or delays is either a value or the TPFG associated with it has redexes.*

From Lem. 13 and 14 it follows that

▶ **Theorem 15.** *Given a circuit $f : 0 \to m, m \neq 0$ in* **ECCirc** *the local rules will always rewrite its TPFG representation in a finite number of steps into a TPFG representation of a* ***value*** $\mathbf{v}$ *such that $f = \mathbf{v}$.*

Finally, it can be shown that the graph rewriting is *efficient.*

▶ **Lemma 16.** *Any rule of the graph rewrite system is applicable in linear time (in the size of the graph).*

3.2 Feedback and delay

We now need to add rules for delays, which may occur in arbitrary places in the circuit, not just in waveforms. For example, a circuit such as $(t \otimes f) \cdot (1 \otimes \delta) \cdot \wedge$, in TFPG representation, does not have any redex because of the delay. Dealing with the delays requires a complex rule which takes into account the presence of the trace. The trace and the delay must be dealt with together because of the following result which allows us to write any circuit in what we will call *global-delay form.* Note Lem. 10 does not hold for combinational circuits with values. However, the following holds:

▶ **Lemma 17.** *For any combinational circuit $f : m \to n$ there exists a passive circuit $\tilde{f}$ such that $f = (m \otimes \mathbf{v}) \cdot \tilde{f}$ for some $\mathbf{v}$.*

We call the application of the transformation in this lemma the *passification* of the circuit.

▶ **Lemma 18.** *Any circuit f in* $\mathbf{OCirc}^*_\delta$ *can be written as $f = \mathrm{Tr}^m((\delta^n \otimes p) \cdot f')$ for some trace-free, delay-free circuit f', $m, n, p \in \mathbb{N}$.*

Trace-Delay. The most complex rule is the unfolding of the global trace, which also handles the delays. First, we need an unfolding axiom for trace from the unfolding axiom for iteration, by expressing trace in terms of iteration. We have seen that the iterator can be expressed in term of trace, but the converse is also possible.

▶ **Proposition 19** ([15])**.** *For any $f : A \otimes X \to A \otimes Y$,*

$$\mathrm{Tr}^A(f) = \mathrm{iter}^{A \otimes Y}((id_A \otimes \mathsf{w}_Y \otimes id_X) \cdot f) \cdot (\mathsf{w}_A \otimes id_Y).$$

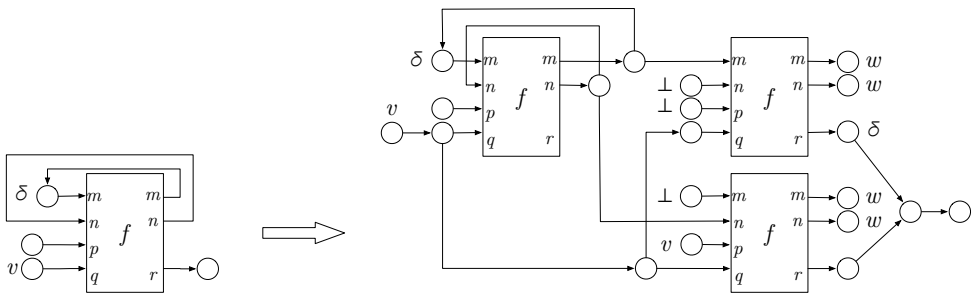
Routine calculations give the following formula for unfolding the trace operator:

▶ **Proposition 20.** *For any $f : A \otimes X \to A \otimes Y$,*

$$\mathrm{Tr}^A(f) = \Delta_X \cdot (\mathrm{iter}^A(f \cdot \mathsf{w}_Y) \otimes id_X) \cdot f \cdot (\mathsf{w}_X \otimes id_Y).$$

Using the above, and the fact that any circuit can be written as a passified (Lem. 17), global-trace, global-delay circuit we can give the following global rewrite rule for circuits with feedback and delays.

▶ **Proposition 21.** *Given a graph representing a passified, global trace, global delay circuit, $f : m + n + p + q \to m + n + r$, the following rewrite rule is sound:*



Proof (Sketch). This rewrite rule is a sequence of valid rewrites. Step (1) represents the unfolding of the trace (Prop. 20). Step (2) uses $\bot$ as the unit of the join-monoid along with the *Unobservable Delay* axiom, to bring the circuit to a form where *Generalised Streaming* (Lem. 10) can be applied, which is step (3). A final simplification removes redundant delays which are not observable (step (5)). A final step (6) restore the global-delay form, using Lem. 18. The resulting circuit can be represented as a TFPG. ◀

The unfolding rule is also efficient:

▶ **Lemma 22.** *A passified, global-trace, global-delay circuit can be unfolded in a time linear in the size of its graph representation.*

We define the overall rewriting system as a cycle of local rewrites until canonical form is reached, followed by trace-delay unfoldings. This system is obviously not terminating, which is consistent with the fact that circuits with feedback can generate infinite waveforms. E.g., $\mathrm{iter}(v :: 1) = v :: \mathrm{iter}(v :: 1) = v :: v :: \mathrm{iter}(v :: 1) = \cdots$.

3.3 Productivity

In a circuit of the form $v :: f = (v \otimes (f \cdot \delta)) \cdot \curlyvee$ value v will be observed *before* whatever the behaviour of f is, since v is *instantaneous* whereas f is guarded by a delay. We call such circuits *productive*, and we add a *labelled* rewrite rule to simplify productive circuit by removing the produced value: $v :: f \overset{v}{\Longrightarrow} f$. This rule is sound because the sub-circuit $v :: -$ can never be part of any redex. So the example above can be written as: $\mathrm{iter}(v :: 1) = v :: \mathrm{iter}(v :: 1) \overset{v}{\Longrightarrow} \mathrm{iter}(v :: 1)$.

However, we note circuits need not be productive in general. There exist circuits where unfoldings never reduce to shape $v :: f$, e.g. $t \cdot \mathrm{iter}(\wedge)$. This is a well known problem caused by a genuine *instant feedback* loop between the output and one of the inputs of the gate. If a circuit has no instant feedback loops, it is guaranteed to be productive.

▶ **Definition 23.** We say that a circuit has *delay-guarded feedback* if its global-delay form is $\mathrm{Tr}^m(\delta^m \cdot f)$.

If a circuit has delay-guarded feedback loops then it is productive. In fact it implements a Mealy automaton.

▶ **Theorem 24.** *Closed delay-guarded circuits with no inputs are productive. Given the TPFG representation of a delay-guarded feedback, the rewrite system will produce a TPFG graph representing a circuit $v :: g$ in a finite number of steps.*

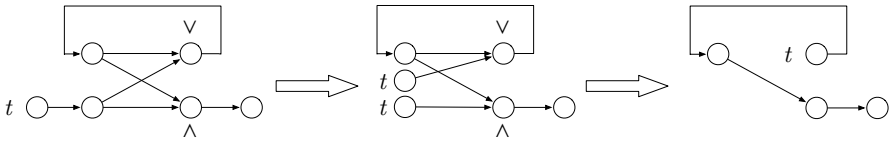
By *closed* above we mean that all inputs to the circuit are provided, i.e. it has type $0 \to m$. Note that the delay-guarded feedback condition is sufficient but not necessary. An interesting example of circuits with non-delay guarded feedback which are productive are the cyclic combinational circuits which we discuss below.

To be able to use the diagrammatic semantics as an operational semantics, we also give a necessary and sufficient non-productivity criterion.

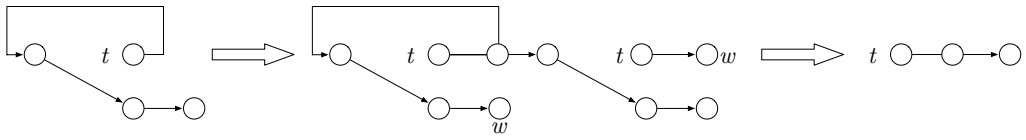
▶ **Theorem 25.** *If a closed, global-trace, global-delay circuit is unproductive after one unfolding then it will always be unproductive.*

3.4 Example: Cyclic combinational circuits

A challenging class of circuits, which are rejected by standard digital design tools, are combinational circuits with feedback which is not delay guarded [26]. Consider Boolean circuits with *and* and *or* gates. Below is an example of such a circuit. Closing the circuit by applying a boolean value at the input (e.g. t) makes it possible to apply the diagrammatic semantics, using the enhanced equational rewrite rules:



There is no rule for "yanking" the superfluous trace, but unfolding the diagram again achieves the same purpose. The circuit then reduces to the constant t, by applying the co-unit and *stub* rules.

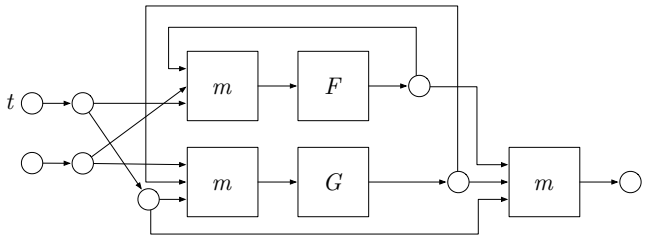


4 Specialising open abstract digital circuits

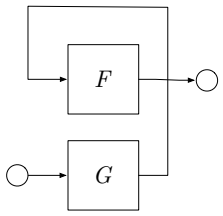
If we are not using the rewrite rules as an operational semantics, and so are not concerned with productivity issues, we can apply the reduction rules to open and to "abstract" circuits, with unspecified components. This gives us a basis for powerful partial evaluation-like optimisations of circuits. This is a new contribution with potentially interesting practical applications.

Consider the circuit represented by the TFPG below, where the gate $m : 3 \to 1$ is a *multiplexer* and F, G are abstract circuits. For readability we omit the input labels of the multiplexer. This circuit, presented in [26], implements the operation *if x then F(G(y)) else G(F(y))*. The circuit has no delays so the feedback loops are combinational, so they are rejected by conventional circuit analysis tools, which disallow instant feedback. However, the multiplexers are set up so that no matter what the value applied at x, the residual circuit is feedback-free. The false feedback loops in the circuit are only a clever way to reuse the two abstract circuits F and G.

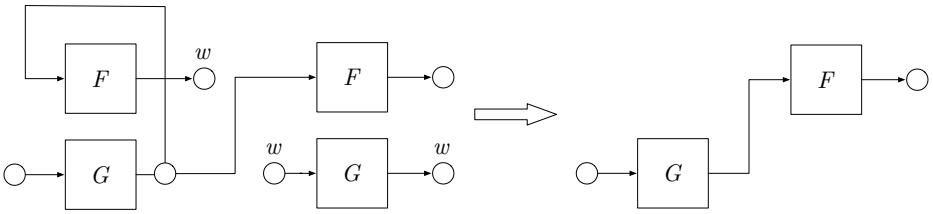
Consider the case when x becomes t, and y is left unspecified:



Routine repeated application of the local rewrite rules for fork, m, and stub results in a circuit which still has a residual feedback loop:



This feedback loop can be yanked, and the circuit is just $G \cdot F$. However, the system does not have a *yank* rule as it would be too expensive to implement, so the unfolding rule is applied again! The *Stub* rule will then remove the first occurrence of F and the second occurrence of G, resulting, as expected, in $G \cdot F$.



To conclude, we would like to emphasise how a circuit that poses a triple challenge to standard digital design tools (open, abstract, combinational feedback) can be partially evaluated in completely automated fashion by our diagrammatic semantics, resulting in a much simpler specialised circuit.

4.1 Pre-logical circuits

We can also model operationally transistor-level circuits, which is also a new capability afforded by the diagrammatic semantics. The circuit framework is general enough to allow operational reasoning about digital circuits at a level of abstraction below logical gates, for example metal-oxide-semiconductor field-effect transistor (MOSFET) circuits. In *saturation mode* such transistors can be considered to take on a discrete set of values which, depending on the circuit and the analysis, can be four-valued (*high impedance* $<$ *high, low* $<$ *unknown.*) or six-valued (*high impedance* $<$ *weak high, weak low*; *weak high* $<$ *strong high*; *weak low* $<$ *strong low*; *strong high, strong low* $<$ *unknown.*). Unlike Boolean logic, where the wire-join construct is not used, in a transistor circuit output wires are joined, and the semantics of the wire-join is that of the value-lattice join operator.

We will work in the six-value lattice $\bot$ (high impedance), h (weak high), H (strong high), l (weak low), L (strong low), $\top$ (unknown). We will take the (idealised) nMOS and pMOS transistors as the basic gates. The nMOS transistor ($\mathsf{n} : 2 \to 1$) works like a low-activated switch, but it only allows low current to flow. High current can flow, but is much diminished. The behaviour of the transistors can be defined equationally in this setting. When implementing a logical gate in MOSFET we want H to correspond to *true* and L to false. The correct behaviour of a gate must keep this representation without, e.g. producing $\top$ or weak output h, l.

Let us now revisit the example of the previous section, but with the multiplexer implemented down to transistors. A very simple circuit is the *inverter*, with which we can build a

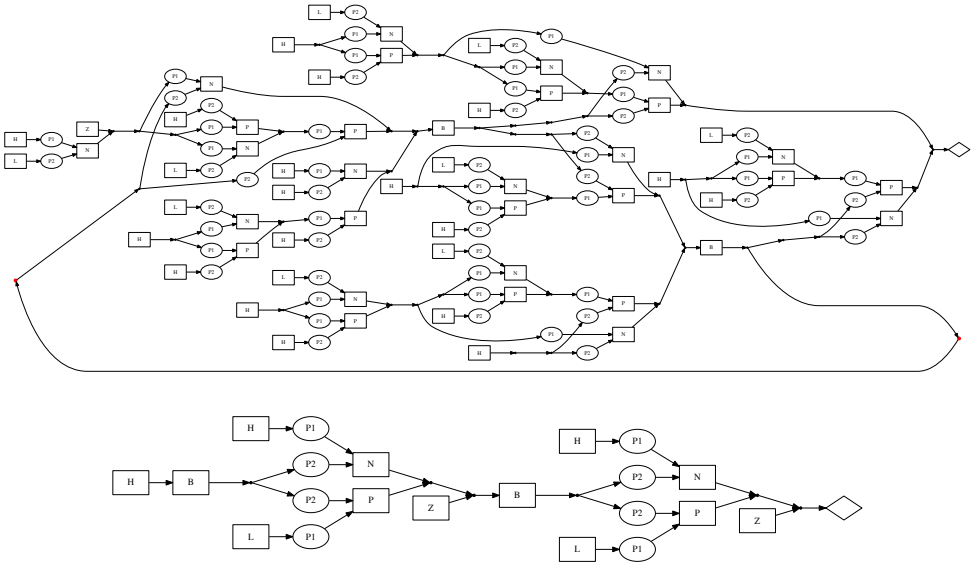


Figure 2 MOSFET circuit and the residual circuit after partial evaluation.

pass-through gate (pass), and the multiplexer (m):

$$\begin{aligned} \mathsf{inv} &= \curlywedge \cdot (1 \otimes \mathsf{h} \otimes 1 \otimes \mathsf{l}) \cdot (\mathsf{p} \otimes \mathsf{n}) \cdot \curlyvee \\ \mathsf{pass} &= \curlywedge^2 \cdot (\mathsf{inv} \otimes 3) \cdot (1 \otimes \mathsf{x} \otimes 1) \cdot (\mathsf{p} \otimes \mathsf{n}) \cdot \curlyvee \\ \mathsf{m} &= (\curlywedge \otimes 2) \cdot (1 \otimes \mathsf{x} \otimes 1) \cdot (2 \otimes \mathsf{inv} \otimes 1) \cdot \mathsf{pass}^2 \cdot \curlyvee. \end{aligned}$$

The abstract circuit from the previous section is represented as a TFPG in the first graph in Fig. 2, and is specialised relative to the abstract circuits (denoted as B in the graph) using a prototype tool[1]. In this case both inputs are provided. The residual circuit is shown as the second graph. It is interesting to note that the MOSFET version of the circuit leads to a different residual circuit compared to the more high level circuit of the previous section. The reason is that reducing the pass-through gates would require more complex rewriting, which cannot be done efficiently in general.

5 Conclusion, related and further work

Some theoretical ingredients we have used in this work have been around for quite a while and it is perhaps somewhat surprising that they have not been put together for a coherent operational and diagrammatic treatment of digital circuits. Our Thm. 6 implies that $\mathbf{OCirc}^*_\delta$ is a Lawvere theory [9] with trace, also known as an iteration theory [10], a concept which has been studied extensively [2], leading to recent connections with rewrite systems [14]. The relation between trace and iteration has also been studied before in a somewhat similar categorical setting [16]. The connection between Lawvere theories and PROPS has also been recently studied [5].

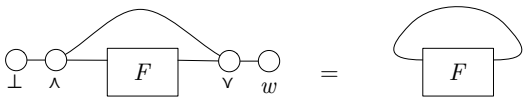
It is also quite surprising that despite major early progress in the algebraic treatment of circuits, [30, 25], this line of work has not come earlier to a systematic conclusion. But the

[1] `https://github.com/AliaumeL/circuit-syntax`

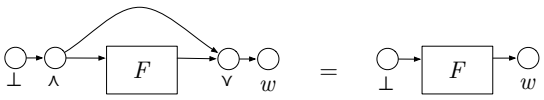
contribution of our work is not merely assembling off-the-shelf components. The *Streaming* axiom is new, and the fact that it generalises to arbitrary passive combinational circuits is a crucial ingredient for our work. To make the unfolding of iteration computationally tractable, the diagrammatic representation required a non-obvious canonical form, which must be easy to compute. Without it our earlier semantics [13] cannot be used as an effective operational semantics.

We have been in particular inspired by the deep connections between monoidal categories and diagrams [29] which *inter alia* have been used in the modelling of quantum protocols [1] and signal-flow graphs [4]. Some contrasts are quite interesting. Unlike in quantum protocols, all digital circuits with no inputs and no outputs are equal whereas in quantum computing they correspond to *scalars*, which allow quantitative aspects to be expressed. Should we have taken a similar direction we could have included quantitative aspects such as power consumption in our formalism, but we would have lost the diagonal property. Obviously, two copies of a circuit will at least sometimes consume more power than one copy.

The signal-flow graphs in [4] are linear and reversible, which is not the case for digital circuits. Without elaborating the mathematics too much, a key difference between their model and ours can be illustrated by the following equality, involving the interaction between fork, join, and disconnected wires, as a trace can be created out of a fork and a join:



Of course, by comparison, in our setting the directionality of the wires never changes, so the correct equality for us is, in contrast:



These two simple diagrammatic equations above truly capture the essential difference between *electric* and *electronic* circuits!

Beyond the scholarly context and technical innovations, we are most excited about the potential applications of our work. Cyclic combinational circuits are a litmus test for circuit modelling theories and we hope the reader can appreciate that in our framework their model is elementary. For comparison, there are few theories that can handle such circuits, and they demand a significant level of mathematical sophistication [27]. The true potential of our method should be the unleashing of symbolic, operational and syntactic methods, such as partial evaluation, for reasoning about and optimising circuits, methods which proved so effective in programming languages.

Acknowledgements. We thank George Constantinides and Alex Smith for feedback and suggestions.

References

1 Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *LICS*, pages 415–425, 2004.

2 Stephen L. Bloom and Zoltán Ésik. *Iteration Theories: The Equational Logic of Iterative Processes.* Springer-Verlag New York, Inc., New York, NY, USA, 1993.

3 Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski, and Fabio Zanasi. Rewriting modulo symmetric monoidal structure. In *LICS*, pages 710–719, 2016.

4 Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. Full abstraction for signal flow graphs. In *POPL*, pages 515–526, 2015.

5 Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. Lawvere categories as composed props. In *CMCS*, pages 11–32, 2016.

6 Virgil Emil Căzănescu and Gheorghe Ştefănescu. Towards a new algebraic foundation of flowchart scheme theory. *Fund. Inf.*, 13(2):171–210, 1990.

7 Charles Consel and Olivier Danvy. Tutorial notes on partial evaluation. In *POPL*, pages 493–501, 1993.

8 Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Löwe. Algebraic approaches to graph transformation – part i: Basic concepts and double pushout approach. In *Handbook of Graph Grammars*, pages 163–246, 1997.

9 Samuel Eilenberg and Jesse B. Wright. Automata in general algebras. *Information and Control*, 11(4):452–470, 1967.

10 Calvin C. Elgot. Monadic computation and iterative algebraic theories. *Studies in Logic and the Foundations of Mathematics*, 80:175–230, 1975.

11 Dan R. Ghica. Diagrammatic reasoning for delay-insensitive asynchronous circuits. In *Computation, Logic, Games, and Quantum Foundations*, pages 52–68, 2013.

12 Dan R. Ghica, Achim Jung, and Aliaume Lopez. Diagrammatic semantics for digital circuits. In *Quantum Physics and Logic (QPL)*, 2017. forthcoming.

13 D. R. Ghica and A. Jung. Categorical semantics of digital circuits. In *Formal Methods in Computer-Aided Design (FMCAD)*, 2016.

14 Makoto Hamana. Strongly normalising cyclic data computation by iteration categories of second-order algebraic theories. In *FCSD*, pages 21:1–21:18, 2016.

15 Masahito Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. Springer Verlag, 1999.

16 Masahito Hasegawa. The uniformity principle on traced monoidal categories. *Electr. Notes Theor. Comput. Sci.*, 69:137–155, 2002.

17 IEEE model standards group. IEEE standard multivalue logic system for VHDL model interoperability (std_logic_1164), May 1993. doi:10.1109/IEEESTD.1993.115571.

18 André Joyal and Ross Street. The geometry of tensor calculus, I. *Adv. in Math.*, 88(1):55–112, 1991.

19 André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. In *Math. Proc. of the Cambridge Phil. Soc.*, volume 119, pages 447–468. Cambridge Univ. Press, 1996.

20 Aleks Kissinger. *Pictures of Processes*. PhD thesis, University of Oxford, 2011. arXiv:1203.0202v2.

21 Robert P. Kurshan and Kenneth L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 10(11):1356–1371, 1991.

22 Stephen Lack. Composing PROPs. *Theory and App. of Categories*, 13(9):147–163, 2004.

23 Peter J. Landin. An abstract machine for designers of computing languages. In *Proc. IFIP Congress*, volume 65, 1965.

24 Charles E. Leiserson and James B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1-6):5–35, 1991.

25 Wayne Luk. Pipelining and transposing heterogeneous array designs. *J. of VLSI Sig. Proc. Sys.*, 5(1):7–20, 1993.

26 Sharad Malik. Analysis of cyclic combinational circuits. In *Proc. IEEE/ACM Int. Conf. on Comp. Aided Design*, pages 618–625, 1993.

27 Michael Mendler, Thomas R. Shiple, and Gérard Berry. Constructive boolean circuits and the exactness of timed ternary simulation. *Form. Meth. Syst. Des.*, 40(3):283–329, 2012.

28 Gordon D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.

29 Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.

30 Mary Sheeran. muFP, A language for VLSI design. In *LISP and Func. Prog.*, pages 104–112, 1984.

31 Alex Simpson and Gordon Plotkin. Complete axioms for categorical fixed-point operators. In *Logic in Computer Science, 2000. Proceedings. 15th Annual IEEE Symposium on*, pages 30–41. IEEE, 2000.

Precongruence Formats with Lookahead through Modal Decomposition

Wan Fokkink[1] and Rob J. van Glabbeek[2]

1 **Vrije Universiteit Amsterdam, The Netherlands**
2 **Data61, CSIRO, Sydney, Australia; and**
University of New South Wales, Sydney, Australia

Abstract

Bloom, Fokkink & van Glabbeek (2004) presented a method to decompose formulas from Hennessy-Milner logic with regard to a structural operational semantics specification. A term in the corresponding process algebra satisfies a Hennessy-Milner formula if and only if its subterms satisfy certain formulas, obtained by decomposing the original formula. They used this decomposition method to derive congruence formats in the realm of structural operational semantics. In this paper it is shown how this framework can be extended to specifications that include bounded lookahead in their premises. This extension is used in the derivation of a congruence format for the partial trace preorder.

1998 ACM Subject Classification F.3.2 [Semantics of Programming Languages] Operational Semantics, F.3.1 [Specifying and Verifying and Reasoning about Programs] Specification Techniques, F.4.1 [Mathematical Logic] Modal Logic, Proof Theory

Keywords and phrases Structural Operational Semantics, Compositionality, Congruence, Modal Logic, Modal Decomposition, Lookahead

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.25

1 Introduction

Structural operational semantics [24] provides specification languages with an interpretation. A transition system specification (TSS), consisting of an algebraic signature and a set of transition rules of the form $\frac{\text{premises}}{\text{conclusion}}$, generates a labelled transition system consisting of transitions between the closed terms over the signature. Transition rules may contain lookahead, meaning that the right-hand side of a premise may occur in the left-hand side of a premise. Lookahead appears for example in the structural operational semantics of a process algebra for process creation [2], an axiomatisation of the process algebra ACP_τ [17], timed LOTOS [22], the stochastic timed process algebra EMPA [3], a probabilistic bisimulation tester [9], and the synchronous programming language Esterel [23]. It also plays an important role in parsing algorithms for e.g. Java [10]. The usefulness of lookahead in formal semantics in the context of agent systems is advocated in [20].

Hennessy-Milner logic (HML) [19] is a dynamic logic to specify properties of a labelled transition system. Larsen [21] showed how to decompose formulas from HML with respect to a TSS in the De Simone format [25]. In [13] this decomposition method was extended to the tyft/tyxt format [18], which allows lookahead. As a step towards this end they used a transformation from [11] to derive so-called "P-ruloids" from a TSS P: transition rules in which the left-hand sides of premises are single variables. A rule has bounded lookahead if all chains of forward dependencies between its premises are finite. In [13] each ruloid with unbounded lookahead is replaced by an equivalent ruloid with bounded lookahead,

by endowing each infinite forward chain with an ordinal count-down. This step is needed because HML, even with infinite conjunctions, cannot capture unbounded lookahead.

An equivalence is a congruence for a given process algebra if it is preserved by all functions in the signature. This is an important property, notably to fit a semantics into an axiomatic framework. Different syntactic formats have been developed for TSSs, to guarantee this property for specific semantics, i.e. for specific behavioural equivalences or preorders. Most of these congruence formats, notably the De Simone format, GSOS [6] and the ready simulation format [15], disallow lookahead.

In [5] the decomposition method for HML is exploited to derive congruence formats, in the context of structural operational semantics, for a wide range of semantics. It takes the ready simulation format as starting point, so the obtained congruence formats are limited to transition rules that have no lookahead. With regard to their congruence format for partial trace semantics, the open question was posed whether the method can be extended to allow some form of lookahead. Doing this is the aim of the current paper.

The key idea in the decomposition method from [5] is that a congruence format $\mathfrak{F}$ for a semantics must ensure that the formulas in a modal characterisation of this semantics are always decomposed into formulas that are again in this modal characterisation. To obtain such a property one needs that if all rules of a TSS P are in $\mathfrak{F}$-format, then so are are all P-ruloids that are needed in the decomposition methods from [13]. However, the ruloids produced by the transformation from unbounded to bounded lookahead in [13] violate most congruence formats, including the partial trace format from [5]. (This is no surprise, as a partial trace format cannot allow unbounded lookahead; see Example 3.)

Lookahead is intrinsically difficult because it establishes an indirect, transitive relation between variables in (proofs using) transition rules. We introduce the notion of a "general" P-ruloid, which intuitively means that different occurrences of the same variable in the rule are linked to each other through its proof; i.e., after renaming some but not all occurrences of a variable x to another variable y, the resulting rule is no longer provable (by means of that proof). We show that the decomposition method from [13] can be restricted to general ruloids. Next we show that general ruloids preserve bounded lookahead. This opens the door to deriving congruence formats that allow bounded lookahead.

As a concrete example we extend the partial trace format from [5] with bounded lookahead. We prove that the resulting partial trace format is preserved by general ruloids, which implies that it is a congruence format for the partial trace preorder. This answers the open question from [5].

Concluding, this paper develops machinery to cope with bounded lookahead in the context of structural operational semantics and modal logic, and applies it to obtain a congruence format for the partial trace preorder. We conjecture that the same machinery also makes it possible to develop congruence formats that allow bounded lookahead for decorated trace and ready simulation semantics, which would provide a positive answer to another open question in [5].

The set-up of the paper is as follows. Section 2 contains technical preliminaries and defines a congruence format for the partial trace preorder that allows bounded lookahead. Section 3 presents the notion of a structured proof, in which different variables are collapsed only if this is required by the proof. A rule is called general if it can be derived by a structured proof. It is shown that each derivable rule is the substitution instance of a general rule. In Section 4 and 5 it is argued that the syntactic restrictions of our partial trace format are inherited by general rules. In Section 6 the developed machinery is used to prove that our partial trace format guarantees that the partial trace preorder induced by a TSS is a congruence. The appendix contains omitted proofs.

2 Preliminaries

This section presents the basic notions of partial trace semantics, modal logic and structural operational semantics, as well as the decomposition method from [13]. It also contains a process algebra for process creation from the literature, which exemplifies the concepts from structural operational semantics and serves as an application of our partial trace format that allows lookahead.

2.1 Hennessy-Milner logic

A *labelled transition system (LTS)* is a pair $(\mathbb{P}, \rightarrow)$ with $\mathbb{P}$ a set of *processes* and $\rightarrow \subseteq \mathbb{P} \times A \times \mathbb{P}$ a *transition relation*, for a set A of *actions*. We call each $(p, a, q) \in \rightarrow$ a *transition*, and write it as $p \xrightarrow{a} q$. A sequence $a_1 \cdots a_n \in A^*$ is a *(partial) trace* of a $p \in \mathbb{P}$ if $p \xrightarrow{a_1} \cdots \xrightarrow{a_n} p'$ for some $p' \in \mathbb{P}$. We write $p \sqsubseteq_T q$ if the set of partial traces of p is included in that of q.

Properties of processes can be formulated in modal logic. Hennessy-Milner logic (HML) [19] characterises the bisimulation equivalence relation on processes. The set $\mathbb{O}$ of *HML formulas* is defined by the BNF grammar $\varphi ::= \bigwedge_{i \in I} \varphi_i \mid \langle a \rangle \varphi \mid \neg \varphi$ where a ranges over A and I is any index set. The satisfaction relation $\models \subseteq \mathbb{P} \times \mathbb{O}$ is defined as usual. In particular, $p \models \langle a \rangle \varphi$ iff $p \xrightarrow{a} p'$ and $p' \models \varphi$ for some $p' \in \mathbb{P}$. We use $\varphi_1 \wedge \varphi_2$ and $\top$ as abbreviations of $\bigwedge_{i \in \{1,2\}} \varphi_i$ and $\bigwedge_{i \in \emptyset} \varphi_i$, respectively. We write $\varphi \equiv \varphi'$ if $p \models \varphi \Leftrightarrow p \models \varphi'$ for each process p in each LTS.

The set $\mathbb{O}_T$ of *partial trace observations* is defined by the BNF grammar $\varphi ::= \top \mid \langle a \rangle \varphi$ where a ranges over A. Given an LTS $(\mathbb{P}, \rightarrow)$, let $\mathcal{O}_T(p)$ denote $\{\varphi \in \mathbb{O}_T \mid p \models \varphi\}$. The class $\mathbb{O}_T^{\equiv}$ denotes the closure of $\mathbb{O}_T$ under $\equiv$.

▶ **Proposition 1** ([16]). *$p \sqsubseteq_T q \Leftrightarrow \mathcal{O}_T(p) \subseteq \mathcal{O}_T(q)$.*

2.2 Transition system specifications

V is an infinite set of variables with $|V| \geq |A|$. Let $var(S)$ denote the set of variables that occur in a syntactic object S. We say that S is *closed* if $var(S) = \emptyset$. A *signature* is a set Σ of function symbols $f \notin V$, with $|\Sigma| \leq |V|$, equipped with an arity function $ar : \Sigma \rightarrow \mathbb{N}$. The set $\mathbb{T}(\Sigma)$ of *terms* over a signature Σ is defined recursively by: $V \subseteq \mathbb{T}(\Sigma)$, and if $f \in \Sigma$ and $t_1, \ldots, t_{ar(f)} \in \mathbb{T}(\Sigma)$ then $f(t_1, \ldots, t_{ar(f)}) \in \mathbb{T}(\Sigma)$. Let $\mathrm{T}(\Sigma)$ denote the set of closed terms over Σ.

A Σ-*substitution* σ is a function from V to $\mathbb{T}(\Sigma)$. Let $\sigma(S)$ denote the syntactic object obtained from S by replacing each occurrence of all $x \in V$ in S by $\sigma(x)$. A Σ-substitution σ is *closed* if $\sigma(x) \in \mathrm{T}(\Sigma)$ for all $x \in V$.

A Σ-*literal* (or *transition*) is an expression $t \xrightarrow{a} t'$ with $t, t' \in \mathbb{T}(\Sigma)$ and $a \in A$. A *transition rule* is of the form $\frac{H}{\alpha}$ with H a set of Σ-literals (the *premises* of the rule) and α a Σ-literal (the *conclusion*). The left- and right-hand side of α are called the *source* and *target* of the rule. A *transition system specification (TSS)* is a pair (Σ, R) with R a collection of transition rules over Σ. The purpose of a TSS (Σ, R) is to specify an LTS $(\mathrm{T}(\Sigma), \rightarrow)$ with as processes the closed terms over Σ and as transition relation a set of closed literals $\rightarrow \subseteq \mathrm{T}(\Sigma) \times A \times \mathrm{T}(\Sigma)$.

Let $P = (\Sigma, R)$ be a TSS. An *irredundant proof* from P of a transition rule $\frac{H}{\alpha}$ is a well-founded, upwardly branching tree in which the nodes are labelled by Σ-literals and some of the leaves are marked "hypothesis", such that: the root is labelled by α, H is the set of labels of the hypotheses, and if β is the label of a node q which is not a hypothesis and K is the set of labels of the nodes directly above q, then $\frac{K}{\beta}$ is a substitution instance of a rule

in R. A *proof* from P of $\frac{K}{\alpha}$ is an irredundant proof from P of $\frac{H}{\alpha}$ with $H \subseteq K$. We note that if a leaf in a proof from P is not a hypothesis, it is a substitution instance of a rule $\frac{\emptyset}{\beta}$ in R.

The proof of $\frac{H}{\alpha}$ is called irredundant because H must equal (instead of include) the set of labels of the hypotheses. Rules that are derived with an irredundant proof may inherit certain syntactic structure from the transition rules in the TSS from which they are derived; in classic proofs this syntactic structure is usually lost, because arbitrary literals can be added as premises of derived rules. Irredundancy of proofs is essential in applications of our decomposition method to derive congruence formats for TSSs [5].

2.3 Syntactic restrictions on transition rules

We present some definitions for transition rules; the majority stems from [18]. A rule is *univariate* if its source does not contain multiple occurrences of the same variable. In a *tytt rule*, the right-hand sides of premises are distinct variables that do not occur in the source. A univariate tytt rule is *tyxt* if its source is a variable, and *tyft* if its source contains exactly one function symbol. A tytt rule is *xytt* if the left-hand sides of its premises are variables. An *xyft rule* is a tyft rule that is also an xytt rule.

The *dependency graph* of a rule with premises $\{t_i \xrightarrow{a_i} t'_i \mid i \in I\}$ is a directed graph with as edges $\{\langle x, y\rangle \mid x \in var(t_i) \text{ and } y \in var(t'_i) \text{ for some } i \in I\}$. A rule is *well-founded* if each backward chain of edges in its dependency graph is finite. It has *lookahead* if there is a variable in the right-hand side of a premise that also occurs in the left-hand side of a premise; the lookahead is *bounded* if each forward chain of edges in the dependency graph is finite.

A variable in a rule is *free* if it occurs in neither the source nor the right-hand sides of premises of this rule. A rule is *pure* if it is well-founded and does not contain free variables.

Each combination of syntactic restrictions on transition rules induces a corresponding syntactic format of the same name for TSSs. For example, a TSS is in *pure tyft/tyxt format* iff it contains only pure tyft and tyxt rules.

2.4 Partial trace format

A preorder is a *precongruence* if, for all functions f in the signature, $p_i \sqsubseteq q_i$ for all $i = 1, \ldots, ar(f)$ implies $f(p_1, \ldots, p_{ar(f)}) \sqsubseteq f(q_1, \ldots, q_{ar(f)})$. Likewise, an equivalence is a *congruence* if it is preserved by all functions in the signature. Here we extend the precongruence format for the partial trace preorder from [5] with bounded lookahead, answering an open question from [5].

Let Λ be a predicate on $\{(f, i) \mid f \in \Sigma, 1 \leq i \leq ar(f)\}$; intuitively it marks those arguments of function symbols that may contain processes that have started running. For example, the first argument of the sequential composition operator from process algebra is typically marked by Λ, but the second argument of this operator generally is not (cf. the process algebra APC in Section 2.5). If $\Lambda(f, i)$, then the argument i of f is called Λ-*liquid*; else it is Λ-*frozen*. An occurrence of a variable x in a term $t \in \mathbb{T}(\Sigma)$ is *[at a]* Λ-*liquid [position]* if either $t = x$, or $t = f(t_1, \ldots, t_{ar(f)})$ and the occurrence of x is Λ-liquid in t_i for some liquid argument i of f. A variable in a tytt rule over Σ is Λ-*floating* if either it occurs as the right-hand side of a premise, or it occurs exactly once in the source, at a Λ-liquid position.

▶ **Definition 2.** Let Λ be a predicate on the arguments of function symbols. A tytt rule is Λ-*partial trace safe* if:

- it has bounded lookahead, and
- each Λ-floating variable has at most one occurrence in total in the left-hand sides of the premises and in the target; this occurrence must be at a Λ-liquid position.

A TSS is in *partial trace format* if it is in tyft/tyxt format and its rules are Λ-partial trace safe with respect to some Λ.

This format extends the partial trace format from [5] in allowing bounded lookahead. By abuse of terminology we reuse the format name from [5] for our more liberal format.

▶ Remark. If a TSS is in partial trace format, then there is a smallest predicate Λ for which all its rules are Λ-partial trace safe; it is *generated* by the second condition above.

This paper develops the machinery to prove that the partial trace preorder induced by a TSS in partial trace format is a precongruence (see Corollary 37). The next example shows that the partial trace format cannot allow unbounded lookahead.

▶ **Example 3.** Let $p \xrightarrow{a} r_i$ and $r_{i+1} \xrightarrow{a} r_i$ for all $i \in \mathbb{Z}_{\geq 0}$, and $q \xrightarrow{a} q$. Clearly $q \sqsubseteq_T p$, as the partial traces of both processes are a^i for all $i \in \mathbb{Z}_{\geq 0}$. Let the unary function symbol f be defined by the xyft rule $\frac{\{x_i \xrightarrow{a} x_{i+1} | i \in \mathbb{Z}_{\geq 0}\}}{f(x_0) \xrightarrow{b} \mathbf{0}}$. where $\mathbf{0}$ is a constant. Then $f(q) \not\sqsubseteq_T f(p)$, because $f(q) \xrightarrow{b} \mathbf{0}$ while $f(p)$ cannot perform a b-transition.

The next example shows that the partial trace format cannot allow multiple occurrences of a Λ-floating variable in left-hand sides of premises.

▶ **Example 4.** Let $p \xrightarrow{a} p'$, $p \xrightarrow{a} p''$, $p' \xrightarrow{b} \mathbf{0}$, and $p'' \xrightarrow{c} \mathbf{0}$. Moreover, let $q \xrightarrow{a} q'$, $q' \xrightarrow{b} \mathbf{0}$, and $q' \xrightarrow{c} \mathbf{0}$. Clearly $q \sqsubseteq_T p$, as the completed traces of both processes are ab and ac. Let the unary function symbol f be defined by the xyft rule $\frac{x \xrightarrow{a} y \quad y \xrightarrow{b} z \quad y \xrightarrow{c} z'}{f(x) \xrightarrow{d} \mathbf{0}}$. Then $f(q) \not\sqsubseteq_T f(p)$, because $f(q) \xrightarrow{d} \mathbf{0}$ while $f(p)$ cannot perform a d-transition.

Since $p \sqsubseteq_T q$ allows that $p \not\xrightarrow{a}$ (i.e., p cannot perform any a-transitions) while $q \xrightarrow{a} q'$, clearly the partial trace format cannot contain so-called negative premises $t \not\xrightarrow{a}$; cf. [5, Example 13 – aliased 11.2]. (For partial trace *equivalence*, Λ-frozen variables can be allowed to occur in negative premises; see [5, Theorem 9 – aliased 11.3].) Moreover, negative premises do not combine well with lookahead; cf. [13, Example 7 – aliased 3.15]. For these reasons the current paper focuses on so-called positive TSSs that do not contain negative premises.

2.5 Application: Algebra for process creation

Baeten & Vaandrager [2] defined a process algebra APC for process creation. Its structural operational semantics contains one transition rule with lookahead. From our congruence format it follows immediately that the partial trace preorder is a precongruence for APC. For simplicity only part of its syntax and semantics is presented here; some auxiliary operators needed for the axiomatisation and the encapsulation operator are omitted.

APC contains the following constants: actions a from a set A, successful termination ε, and deadlock δ. The rules are:

$$\frac{}{a \xrightarrow{a} \varepsilon} \qquad \frac{}{\varepsilon \xrightarrow{\surd} \delta}$$

Let e range over $A \cup \{\surd\}$. The two rules for the binary alternative composition operator $+$ are:

$$\frac{x_1 \xrightarrow{e} y_1}{x_1 + x_2 \xrightarrow{e} y_1} \qquad \frac{x_2 \xrightarrow{e} y_2}{x_1 + x_2 \xrightarrow{e} y_2}$$

The asymmetric binary parallel composition $\between$ assumes a partially defined communication function $| : A \times A \to A$; it passes through a termination signal $\surd$ from the right side only.

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 \between x_2 \xrightarrow{a} y_1 \between x_2} \qquad \frac{x_2 \xrightarrow{e} y_2}{x_1 \between x_2 \xrightarrow{e} x_1 \between y_2} \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{b} y_2 \quad a|b = c}{x_1 \between x_2 \xrightarrow{c} y_1 \between y_2}$$

The second and third rule rule for the binary sequential composition operator $\cdot$ below are unusual: they spawn a parallel component. The last rule contains lookahead.

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 \cdot x_2 \xrightarrow{a} y_1 \cdot x_2} \qquad \frac{x_1 \xrightarrow{\surd} y_1 \quad x_2 \xrightarrow{e} y_2}{x_1 \cdot x_2 \xrightarrow{e} y_1 \between y_2} \qquad \frac{x_1 \xrightarrow{\surd} y_1 \quad y_1 \xrightarrow{a} z_1 \quad x_2 \xrightarrow{b} y_2 \quad a|b=c}{x_1 \cdot x_2 \xrightarrow{c} z_1 \between y_2}$$

Finally, the unary operator **new**, originating from [1], creates a process that can be put in parallel with an existing process, in view of the peculiar semantics of sequential composition.

$$\frac{}{\mathbf{new}(x) \xrightarrow{\surd} x \cdot \delta} \qquad \frac{x \xrightarrow{a} y}{\mathbf{new}(x) \xrightarrow{a} \mathbf{new}(y)}$$

These are all tyft rules, because in each rule the source contains a single function symbol and the variables in the source and in the right-hand sides of the premises are all distinct. Furthermore, these rules clearly all have bounded lookahead. We take the arguments of $\between$ and **new** and the first argument of $\cdot$ to be Λ-liquid, and the arguments of $+$ and the second argument of $\cdot$ to be Λ-frozen. Each Λ-floating variable has at most one occurrence in total in the left-hand sides of the premises and in the target; this occurrence is in all cases at a Λ-liquid position. Hence the TSS is in the partial trace format. The transition rules for the omitted operators are also in this format. So the partial trace preorder is a precongruence with regard to APC.

2.6 Decomposition of HML formulas

In [13] it was shown how to decompose HML formulas with respect to process terms, given a TSS in tyft/tyxt format. The decomposition method uses a collection of pure xytt rules extracted from this TSS, called *ruloids*. We require that there is a proof of a transition $p \xrightarrow{a} q$, with p a closed substitution instance of a term t, iff there exists a proof that uses at the root a ruloid with source t.

▶ **Definition 5.** A collection $\mathcal{R}$ of pure xytt rules is called *a suitable set of ruloids* for a TSS $P = (\Sigma, R)$ if for each $t \in \mathbb{T}(\Sigma)$, $p \in \mathrm{T}(\Sigma)$ and closed substitution σ, the transition $\sigma(t) \xrightarrow{a} p$ is provable from P iff there are a ruloid $\frac{H}{t \xrightarrow{a} u} \in \mathcal{R}$ and a closed substitution σ' where $\sigma'(\alpha)$ is provable from P for all $\alpha \in H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$.

Let $\mathcal{R}$ be a collection of ruloids with bounded lookahead that is suitable for a TSS P. The following definition from [13] assigns to each term $t \in \mathbb{T}(\Sigma)$ and each observation $\varphi \in \mathbb{O}$ a collection $t^{-1}_{\mathcal{R}}(\varphi)$ of decomposition mappings $\psi : V \to \mathbb{O}$. A closed substitution instance $\sigma(t)$ satisfies φ iff for some $\psi \in t^{-1}_{\mathcal{R}}(\varphi)$, $\sigma(x)$ satisfies the formula $\psi(x)$ for all $x \in var(t)$.

▶ **Definition 6.** Let $\mathcal{R}$ be a collection of ruloids with bounded lookahead, suitable for a TSS $P = (\Sigma, R)$. Then $\cdot^{-1}_{\mathcal{R}} : \mathbb{T}(\Sigma) \to (\mathbb{O} \to \mathcal{P}(V \to \mathbb{O}))$ is defined by:

- $\psi \in t^{-1}_{\mathcal{R}}(\langle a \rangle \varphi)$ iff there is a ruloid $\frac{H}{t \xrightarrow{a} u} \in \mathcal{R}$ and a $\chi \in u^{-1}_{\mathcal{R}}(\varphi)$ such that $\psi : V \to \mathbb{O}$ is given by

$$\psi(x) = \begin{cases} \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y) \ \wedge \ \chi(x) & \text{if } x \in var(u) \\ \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y) & \text{if } x \notin var(u). \end{cases}$$

- $\psi \in t^{-1}_{\mathcal{R}}(\bigwedge_{i \in I} \varphi_i)$ iff $\psi(x) = \bigwedge_{i \in I} \psi_i(x)$ where $\psi_i \in t^{-1}_{\mathcal{R}}(\varphi_i)$ for all $i \in I$.

- $\psi \in t_{\mathcal{R}}^{-1}(\neg\varphi)$ iff there is a function $h : t_{\mathcal{R}}^{-1}(\varphi) \to var(t)$ such that $\psi : V \to \mathbb{O}$ is given by $\psi(x) = \bigwedge_{\chi \in h^{-1}(x)} \neg\chi(x)$.

This recursive definition is well-founded because the ruloid employed in the case $t_{\mathcal{R}}^{-1}(\langle a\rangle\varphi)$ has bounded lookahead.

We note that, in contrast to the setting of [5] without lookahead, $x \notin var(t)$ here does not imply $\psi(x) \equiv \top$.

▶ **Example 7.** Consider the TSS of xyft rules $\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{b} g(y)}$ and $\frac{x \xrightarrow{c} y \quad y \xrightarrow{d} z}{g(x) \xrightarrow{e} \mathbf{0}}$. Let the suitable collection of ruloids $\mathcal{R}$ contain $\frac{y \xrightarrow{a} x}{f(y) \xrightarrow{b} g(x)}$ and $\frac{x \xrightarrow{c} y \quad y \xrightarrow{d} z}{g(x) \xrightarrow{e} \mathbf{0}}$.

We calculate a $\psi \in f(y)_{\mathcal{R}}^{-1}(\langle b\rangle\langle e\rangle\top)$. For a start, the ruloid $\frac{y \xrightarrow{a} x}{f(y) \xrightarrow{b} g(x)}$ yields $\psi(y) = \langle a\rangle\psi(x)$ and $\psi(x) = \chi(x)$ for some $\chi \in g(x)_{\mathcal{R}}^{-1}(\langle e\rangle\top)$. For the calculation of χ we use the ruloid $\frac{x \xrightarrow{c} y \quad y \xrightarrow{d} z}{g(x) \xrightarrow{e} \mathbf{0}}$. This yields $\chi(x) = \langle c\rangle\chi(y)$ and $\chi(y) = \langle d\rangle\chi(z)$ and $\chi(z) = \top$. Concluding, $\psi(y) = \langle a\rangle\langle c\rangle\langle d\rangle\top$.

The syntactic overloading of y, i.e. its occurrence in both the first and second ruloid, underlines the importance of the separate case in the definition of $\psi \in t_{\mathcal{R}}^{-1}(\langle a\rangle\varphi)$ in Definition 6, for $x \notin var(u)$. Else we would get $\psi(y) = \langle a\rangle\psi(x) \wedge \chi(y)$, yielding a spurious conjunct $\langle d\rangle\top$ for $\psi(y)$.

We reformulate the decomposition result from [13].

▶ **Theorem 8.** *Let $\mathcal{R}$ be a collection of ruloids with bounded lookahead, suitable for a TSS $P = (\Sigma, R)$. Then for each $t \in \mathbb{T}(\Sigma)$, $\sigma : V \to \mathrm{T}(\Sigma)$ and $\varphi \in \mathbb{O}$:*

$$\sigma(t) \models \varphi \quad \Leftrightarrow \quad \exists \psi \in t_{\mathcal{R}}^{-1}(\varphi)\, \forall x \in var(t) : \sigma(x) \models \psi(x)$$

In [13] P was required to be in tyft/tyxt format, and a specific collection $\mathcal{R}$ was constructed. However, the proof of Theorem 8 only uses that $\mathcal{R}$ has the property of Definition 5. The requirement that P be in tyft/tyxt format was needed merely to ensure that such an $\mathcal{R}$ can be found.

2.7 Construction of ruloids

We briefly sketch the extraction of ruloids from a TSS P in tyft/tyxt format, as employed in [13]. First, employing a conversion from [18], if the source of a rule is of the form x then this variable is replaced by a term $f(x_1, \ldots, x_{ar(f)})$ for each $f \in \Sigma$. This yields an intermediate TSS $P^\dagger$ in tyft format, of which all rules are provable from P. Next, using a construction from [11], the left-hand sides of premises are reduced to variables. Roughly the idea is, given a premise $f(t_1, \ldots, t_{ar(f)}) \xrightarrow{a} y$ in a rule r, and a rule $\frac{H}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{a} t}$, to transform r by replacing the aforementioned premise by H, y by t, and the x_i by the t_i; this is repeated (transfinitely) until all premises with a non-variable left-hand side have disappeared. Each infinite sequence of such substitutions converges to an infinite sequence of variable replacements; these variables are unified. The result is a TSS $P^\ddagger$ in xyft format, of which all rules are provable from $P^\dagger$ [11]. Next, the premises for which there is no backward chain in the dependency graph to a variable in the source are eliminated, by substituting closed terms for the variables in such premises. The resulting TSS in pure xyft format is denoted by P^+; its rules are provable from $P^\ddagger$ [11]. By [13, Proposition 3 – aliased 3.4] the pure xytt rules irredundantly provable from P^+ constitute a suitable collection of ruloids for P.

▶ **Example 9.** Consider the process algebra APC from Section 2.5, with $A = \{a, b, c, d, e\}$ and a communication function that includes $a|b = c$ and $c|d = e$. The next ruloid can be derived using the third rule for parallel composition and the third rule for sequential composition.

$$\frac{x_1 \xrightarrow{\surd} y_1 \quad y_1 \xrightarrow{a} z_1 \quad x_2 \xrightarrow{b} y_2 \quad x_3 \xrightarrow{d} y_3}{(x_1 \cdot x_2) \between x_3 \xrightarrow{e} (z_1 \between y_2) \between y_3}$$

Using [11, Lemma 2.10], it follows that these ruloids are provable from P. Hence another suitable collection of ruloids is given by *all* pure xytt rules provable from P, or all pure xytt rules *irredundantly* provable from P. In Section 3 we will define *P-general ruloids* such that each pure xytt rule irredundantly provable from P is a substitution instance of a P-general ruloid with the same source. This implies that the collection of P-general ruloids is suitable.

In [13] an additional step in the construction of ruloids was made to ensure that they all have bounded lookahead. Each ruloid with unbounded lookahead was replaced by an equivalent ruloid with bounded lookahead, by endowing each infinite forward chain with an ordinal count-down. However, the ruloids produced by this step violate most congruence formats. (A notable exception is the full tyft/tyxt format, as a congruence format for bisimulation semantics; see [13, Corollary 1 – aliased 4.1].) In particular, starting with a TSS in partial trace format, this step produces ruloids in which Λ-floating variables may have multiple occurrences in left-hand sides of premises. This is no surprise, as in Example 3 it was shown that the partial trace format must exclude unbounded lookahead. Here we avoid this additional step by considering only TSSs in tyft/tyxt format with bounded lookahead. We will prove in Section 4 that for such TSSs P, each P-general ruloid has bounded lookahead.

3 Structured proofs and general rules

The following example shows that the second condition of the partial trace format is not always preserved by irredundant proofs of pure xytt rules.

▶ **Example 10.** Consider the TSS with bounded lookahead consisting of the xyft rules

$$\frac{\{y_{i+1} \xrightarrow{a} y_i \mid i \in \mathbb{Z}_{\geq 0}\}}{f(x) \xrightarrow{b} g(x, y_0)} \qquad \frac{}{x \xrightarrow{a} x} \qquad \frac{x \xrightarrow{c} y}{f(x) \xrightarrow{d} f(y)}$$

In view of the third rule, the argument of f is Λ-liquid. So by the first rule, the arguments of g are Λ-liquid as well. Clearly the TSS is in partial trace format.

Substituting z for x and for all y_i in the first rule as well as for x in the second rule, we can derive the rule $\frac{}{f(z) \xrightarrow{b} g(z,z)}$. The two occurrences of the Λ-floating variable z in the target violate the partial trace format.

This counter-example is spurious: the derived rule is a substitution instance of the rule $\frac{}{f(z) \xrightarrow{b} g(z,z')}$ with $z \neq z'$, which does adhere to the partial trace format. The latter rule can be derived in a similar fashion, by substituting z' (instead of z) for all y_i in the first rule as well as for x in the second rule. The irredundant proof of $\frac{}{f(z) \xrightarrow{b} g(z,z)}$ is not "general": there is no need to replace the two arguments of g in the target by the same variable. On the other hand, the same variable must be substituted for all the y_i, so that the premises of the first rule can be derived by the second rule in the TSS.

We will show that a suitable collection of ruloids is formed by the so-called "general" pure xytt rules, which are derived by an irredundant proof in which terms $\sigma(x)$ and $\sigma(y)$ with

$x \neq y$ only have variables in common if this is imposed by the proof. For example, let the TSSs P_1 and P_2 both contain the rule $\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{b} y}$, while P_1 contains $\frac{}{g(x) \xrightarrow{a} x}$ and P_2 contains $\frac{}{g(x) \xrightarrow{a} y}$. The rule $\frac{}{f(g(z)) \xrightarrow{b} z}$ is irredundantly provable from both P_1 and P_2; however, this rule is P_1-general but not P_2-general. In contrast, the rule $\frac{}{f(g(z)) \xrightarrow{b} z'}$ is P_2-general, but not provable from P_1.

In Section 4 and 5 it will be shown that general rules do preserve the partial trace format. To formally define the notion of a general rule, we first provide an alternative characterisation of (irredundant) provability, roughly following [11]. We will consider irredundant proofs with minimal variable unifications. However, irredundant proofs abstract away from the variables that are being unified; in contrast, proof structures contain variable unifications explicitly.

▶ **Definition 11.** Let $\pi = (B, \alpha, \varphi)$ where:

- B is a set of transition rules which do not have any variables in common,
- α is a literal of the form $s_\pi \xrightarrow{a} w$ with $s_\pi \in \mathbb{T}(\Sigma)$ and $w \in V$ where $var(\alpha) \cap var(B) = \emptyset$ and $w \notin var(s_\pi)$, and
- φ is an injective mapping from B to $\{\alpha\} \cup \{\beta \mid \beta$ a premise of a rule in $B\}$, such that
  - the conclusion of b and $\varphi(b)$ carry the same action for all $b \in B$, and
  - all chains $b_0, b_1, b_2, \ldots$ in B with each $\varphi(b_{i+1})$ a premise of b_i are finite.

In the sequel, the *premises* of π are α and the premises of rules in B, and $top(\pi)$ denotes the collection of premises of π that are outside the image of φ.

A rule $b_0 \in B$, or a premise of b_0, is said to be *above* a premise β if there exists a chain $b_1, \ldots, b_n$ in B with $\varphi(b_i)$ a premise of b_{i+1} for all $0 \leq i < n$ and $\varphi(b_n) = \beta$.

π is a *proof structure* if each rule in B is above α. It is a proof structure *over* a TSS $P = (\Sigma, R)$ if each rule in B is in R modulo alpha-conversion (i.e., renaming of variables).

A substitution σ *matches* π if $\sigma(s_\pi) = s_\pi$ and, for all $b \in B$, $\sigma(conclusion(b)) = \sigma(\varphi(b))$.

▶ **Proposition 12.** *A rule $\frac{H}{\gamma}$ is provable from a TSS P iff there exists a proof structure $\pi = (B, \alpha, \varphi)$ over P and a substitution σ that matches π, such that $\sigma(top(\pi)) \subseteq H$ and $\sigma(\alpha) = \gamma$. It is irredundantly provable if $\sigma(top(\pi)) = H$.*

Proof. Given a proof structure $\pi = (B, \alpha, \varphi)$ and a matching substitution σ, an irredundant proof of $\sigma(\frac{top(\pi)}{\alpha})$ is obtained as the (multi)set of premises of π, each premise β labelled by $\sigma(\beta)$, ordered into a tree by the "above" relation; $top(\pi)$ will be the set of "hypotheses".

Conversely, each irredundant proof π of a rule $\frac{H}{t \xrightarrow{a} u}$ can be converted into a proof structure (B, α, φ) by replacing each non-hypothesis node in π by an incarnation of the transition rule applied in that node, where, using alpha-conversion, all incarnations are given disjoint sets of variables. Take $\alpha := (t \xrightarrow{a} w)$ for a fresh variable w. When the rules for each two nodes have disjoint variable sets, the substitutions used in all nodes can be united into one substitution matching the entire proof structure.

One point of concern in the above construction is whether there are enough variables to allocate a disjoint set of variables to the rules for each node in π. As V is infinite, this constraint is satisfied if the number of nodes in π is not larger than $|V|$, which is the case if the branching degree of π, i.e. the number of premises in each rule, is no larger than $|V|$. In [11] this was achieved by means of a requirement on TSSs, namely of being "small". Here we just make sure that the set of *all* literals is not larger than $|V|$. This is a simple consequence of our requirements that $|\Sigma| \leq |V|$ and $|A| \leq |V|$ (cf. Lemma 6 – aliased 6.4 – in [5]). ◀

A different proof of a small variation of this characterisation can be found in [11].

▶ **Example 13.** Consider the TSS in Example 10. As running example in this section we introduce a proof structure of the following shape, where downward arrows depict φ.
One matching substitution σ_1 maps w to $g(z,z)$ and all other variables to z. Another matching substitution σ_2 maps w to $g(z,z')$, x to z and all other variables to z'.

A variable x in a proof structure π is *prime* if there exists a matching substitution σ for π with $\sigma(x)$ a variable. The relation $\sim_\pi$ relates those prime variables that are mapped to the same term by each matching substitution for π.

$$\dfrac{\begin{array}{cc} \cdots & x_1 \xrightarrow{a} x_1 \quad x_0 \xrightarrow{a} x_0 \\ & \downarrow \qquad\quad \downarrow \\ \cdots & y_2 \xrightarrow{a} y_1 \quad y_1 \xrightarrow{a} y_0 \end{array}}{\begin{array}{c} f(x) \xrightarrow{b} g(x,y_0) \\ \downarrow \\ f(z) \xrightarrow{b} w \end{array}}$$

▶ **Definition 14.** Let $\pi = (B, \alpha, \varphi)$ be a proof structure. Let $\sim_\pi$ be the least equivalence relation on $\mathbb{T}(\Sigma)$ satisfying:
- if $b = \frac{H}{t \xrightarrow{a} u} \in B$ and $\varphi(b) = (t' \xrightarrow{a} u')$ then $t \sim_\pi t'$ and $u' \sim_\pi u$, and
- if $f(t_1, \ldots, t_k) \sim_\pi f(u_1, \ldots, u_k)$ then $t_i \sim_\pi u_i$ for all $i = 1, \ldots, k$.

A variable $x \in var(\pi)$ is *composite* if $x \sim_\pi t$ with $t \notin V$, and *prime* otherwise.

▶ **Observation 15.** *A substitution σ matches π iff $\sigma(s_\pi) = s_\pi$ and $\sigma(t) = \sigma(u)$ for all terms $t, u \in \mathbb{T}(\Sigma)$ with $t \sim_\pi u$.*

▶ **Example 16.** For the proof structure in Example 13, $x_i \sim_\pi y_i$ and $x_i \sim_\pi y_{i+1}$ for all $i \in \mathbb{Z}_{\geq 0}$. Moreover, $x \sim_\pi z$ and $w \sim_\pi g(x, y_0)$. So the two equivalence classes of prime variables modulo $\sim_\pi$ are $\{x_i, y_i \mid i \in \mathbb{Z}_{\geq 0}\}$ and $\{x, z\}$.

A substitution is *minimal* for a proof structure if it is matching and provides as little syntactic structure to (substitution instances of) variables as possible, and induces as few identifications of variables as possible.

▶ **Definition 17.** A substitution ρ for a proof structure π is *minimal* if:
- $\rho(x) = x$ for each $x \in var(s_\pi)$ and $\rho(x) \in V$ for each prime variable $x \in var(\pi)$,
- $\rho(x) = \rho(y)$ iff $x \sim_\pi y$, for each pair of prime variables $x, y \in var(\pi)$, and
- $\rho(t) = \rho(u)$ for each two terms $t, u \in \mathbb{T}(\Sigma)$ with $t \sim_\pi u$.

A rule r is *P-general* if there exists a proof structure $\pi = (B, \alpha, \varphi)$ over P and a substitution ρ that is minimal for π such that $r = \rho(\frac{top(\pi)}{\alpha})$. The pair (π, ρ) is called a *structured proof* of r from P.

▶ **Example 18.** The first matching substitution σ_1 for the proof structure in Example 13 is not minimal, because it maps the variables in the two equivalence classes modulo $\sim_\pi$ to the same variable z.

The second matching substitution σ_2 for this proof structure is minimal, meaning that the rule $\frac{}{f(z) \xrightarrow{b} g(z,z')}$ is general with regard to the TSS in Example 10.

The following proposition is a pivotal result for this paper.

▶ **Proposition 19.** *A rule is irredundantly provable from a TSS P iff it is a substitution instance of a P-general rule with the same source.*

Proof. $\boxed{\Leftarrow}$ Let the rule r be a substitution instance of a rule $\rho(\frac{top(\pi)}{s_\pi \xrightarrow{a} w})$ with the same source, where $\pi = (B, s_\pi \xrightarrow{a} w, \varphi)$ is a proof structure over P and ρ a minimal substitution for π. Then $r = \sigma(\rho(\frac{top(\pi)}{s_\pi \xrightarrow{a} w}))$ for some substitution σ. By assumption, $\sigma(\rho(s_\pi)) = \rho(s_\pi) = s_\pi$. By Observation 15 and the third requirement on minimal substitutions, ρ matches π. Therefore, also $\sigma \circ \rho$ matches π, so by Proposition 12 r is irredundantly provable from P.

$\boxed{\Rightarrow}$ Let the rule r be irredundantly provable from P. By Proposition 12 $r = \sigma(\frac{top(\pi)}{s_\pi \xrightarrow{a} w})$ for some proof structure $\pi = (B, s_\pi \xrightarrow{a} w, \varphi)$ and a matching substitution σ. We will now construct a substitution ρ that is minimal for π, and a substitution ν with $\sigma = \nu \circ \rho$. This immediately yields the required result.

For each $\sim_\pi$-equivalence class C of prime variables we pick a $y_C \in C$ and take $\rho(x) := y_C$ for all $x \in C$ – if possible we choose $y_C \in var(s_\pi)$. This way the first two requirements of a minimal substitution are met. In particular, if $x, y \in var(s_\pi)$ with $x \sim_\pi y$ then $\sigma(x) = x$ and $\sigma(y) = y$, which implies that x and y are prime, and by Observation 15 $x = \sigma(x) = \sigma(y) = y$; thus $\rho(x) = x$. Moreover, take $\nu(y_C) := \sigma(y_C)$, so that $\sigma(x) = \sigma(y_C) = \nu(y_C) = \nu(\rho(x))$ for all $x \in C$, using Observation 15. The substitution ν satisfies $\nu(z) = z$ for all other variables z.

With structural induction on $\sigma(x)$ we proceed to define $\rho(x)$ for composite variables $x \in var(\pi)$, such that $\sigma(x) = \nu(\rho(x))$. Simultaneously, with structural induction on $\sigma(t)(= \sigma(u))$, we establish $\rho(t) = \rho(u)$ for each pair of terms t, u with $t \sim_\pi u$.

Let $t, u \notin V$ be terms with $t \sim_\pi u$. Let $t = f(t_1, \ldots, t_k)$ and $u = g(u_1, \ldots, u_m)$. By Observation 15 $\sigma(t) = \sigma(u)$, so $f = g$ and $k = m$. By Definition 14 $t_i \sim_\pi u_i$, so by induction $\rho(t_i) = \rho(u_i)$, for all $i = 1, \ldots, k$, and hence $\rho(t) = \rho(u)$.

Now let $x \in var(\pi)$ be composite; say $x \sim_\pi t$ for some term $t \notin V$. By Observation 15 $\sigma(x) = \sigma(t)$, so for each $y \in var(t)$ the term $\sigma(y)$ is a proper subterm of $\sigma(x)$. By induction, $\rho(y)$ has already been defined before we get to defining $\rho(x)$, and $\sigma(y) = \nu(\rho(y))$. Hence $\rho(t)$ is well-defined, and $\sigma(t) = \nu(\rho(t))$, so we can take $\rho(x) := \rho(t)$, thereby obtaining $\sigma(x) = \nu(\rho(x))$. By the argument above, this definition is independent of the choice of t.

Finally, if $x \sim_\pi y$ and one of these variables is composite, then both are composite and $x \sim_\pi t \sim_\pi y$ for some term $t \notin V$. Now $\rho(x) = \rho(y)$ follows by transitivity. ◀

▶ **Example 20.** With regard to the TSS in Example 10, the irredundantly provable rule $\frac{}{f(z) \xrightarrow{b} g(z,z)}$ is a substitution instance of the general rule $\frac{}{f(z) \xrightarrow{b} g(z,z')}$.

Now we define a *P-general ruloid* as a P-general pure xytt rule. It follows from Proposition 19 that each irredundantly provable pure xytt rule is a substitution instance of a P-general ruloid with the same source, so the P-general ruloids form a suitable collection of ruloids for P.

We now consider TSSs in univariate tytt format; these syntactic restrictions are part of all congruence formats in the literature. The following definition makes relations between different occurrences of a variable z in a structured proof explicit. The underlying idea of its first case is that syntactic structure is inherited in an upward fashion at the left-hand side of each branch of a proof, and in a downward fashion at the right-hand side. The second case expresses that occurrences of a variable x in a rule in the proof inherit syntactic structure from the (unique) occurrence of x in the source or in a right-hand side of a premise of this rule. The third case extends this relationship to variables that are free in a rule in the proof: all occurrences of a free variable x are syntactically linked to each other.

▶ **Definition 21.** Let (π, ρ) with $\pi = (B, \alpha, \varphi)$ be a structured proof from a TSS in univariate tytt format. An *occurrence* of a variable z in this proof is represented by a triple $\theta = (b, \iota, \eta)$ with either $b \in B$ or $b = \alpha$, ι an occurrence (i.e. position) of a variable x in b, and η an

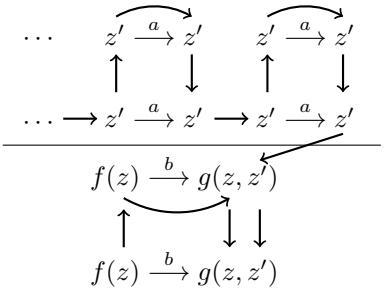
occurrence of z in $\rho(x)$. Sometimes we address such an occurrence as $\langle b, \iota_t, \eta\rangle$ where ι_t is an occurrence of a *term* t in b, and η an occurrence of z in $\rho(t)$.

The relations $\to_z$ and $\leftrightsquigarrow_z$ between the occurrences of z in (π, ρ) are given by:

- if $b = \frac{H}{t \xrightarrow{a} u} \in B$ and $\varphi(b) = (t' \xrightarrow{a} u')$, writing b' for the rule (or α) containing the premise $\varphi(b)$ and ι_t, $\iota_{t'}$, ι_u and $\iota_{u'}$ for the indicated occurrences of t in b, t' in b', u in b and u' in b', respectively, then $\langle b', \iota_{t'}, \eta\rangle \to_z \langle b, \iota_t, \eta\rangle$ for any occurrence η of z in $\rho(t') = \rho(t)$, and $\langle b, \iota_u, \eta'\rangle \to_z \langle b', \iota_{u'}, \eta'\rangle$ for any occurrence η' of z in $\rho(u) = \rho(u')$;
- if $b \in B$ and η is an occurrence of z in $\rho(x)$ for some $x \in var(b)$, then $(b, \iota, \eta) \to_z (b, \iota', \eta)$ where ι is an occurrence of x either in the source of b or in the right-hand side of a premise of b, and ι' is an occurrence of x in the left-hand side of a premise or in the target of b;
- if η is an occurrence of z in $\rho(x)$ with $x \in var(b)$, and either $b = \alpha$, or $b \in B$ and x occurs neither in the source of b nor in the right-hand sides of its premises, then $(b, \iota, \eta) \leftrightsquigarrow_z (b, \iota', \eta)$ for ι and ι' any two different occurrences of x.

Let $\sim_z$ denote the smallest equivalence relation containing $\to_z \cup \leftrightsquigarrow_z$.

▶ **Example 22.** Consider the structured proof from Example 13, after applying the matching substitution σ_2 to it. The relations $\to_z$ and $\to_{z'}$ are depicted by arrows. (The arrows depicting φ have been omitted here.) Relations between different rules (i.e., the vertical ones) are due to the first case of Definition 21, while relations within one rule are due to the second case of Definition 21. Since the TSS in Example 10 does not contain free variables, the third case of Definition 21 does not apply.



We partition the variable occurrences in a rule r into three types: we speak of an *incoming* occurrence if it occurs in the source of r, or in the right-hand side of a premise; an *upwards outgoing* occurrence if it occurs in the left-hand side of a premise; and a *downwards outgoing* occurrence if it occurs in the target of r. This applies to rules $\rho(b)$ associated to a structured proof (π, ρ) with $\pi = (B, \alpha, \varphi)$ and $b \in B$; it also applies to $\rho(\alpha)$ by considering this literal to be a premise. This terminology is motivated by the following observation on the above constructed graph of occurrences of a variable z in a structured proof (π, ρ).

▶ **Observation 23.** *If $(b, \iota, \eta) \to_z (b', \iota', \eta')$ then either*

- *$b' = b$, ι is an incoming occurrence and ι' an (upwards or downwards) outgoing one, or*
- *$\varphi(b')$ is a premise of b, ι is an upwards outgoing occurrence in b, and ι' is an incoming occurrence in b', or*
- *$\varphi(b)$ is a premise of b', ι is a downwards outgoing occurrence in b, and ι' is an incoming occurrence in b'.*

▶ **Example 24.** Consider the arrows in the picture in Example 22 that depict the relations $\to_z$ and $\to_{z'}$. The upward arrows are from an upwards outgoing to an incoming occurrence of z or z', the downward arrows from a downwards outgoing to an incoming occurrence of z or z', the straight horizontal arrows from an incoming to an upwards outgoing occurrence of

z', and the diagonal and curved horizontal arrows from an incoming to a downwards outgoing occurrence of z or z'.

Note that an occurrence $\theta = (b, \iota, \eta)$ of a variable z in a structured proof (π, ρ) refers to an occurrence of z in the rule $\rho(b)$; θ is called *incoming* or *outgoing*, or Γ-*liquid*, for a given predicate Λ on arguments of function symbols, iff the referred occurrence of z in $\rho(b)$ has these properties. Note that θ is *incoming* or *outgoing* iff ι is such an occurrence.

▶ **Observation 25.** *For each outgoing occurrence θ of z in a structured proof (π, ρ) there is at most one incoming occurrence θ' of z with $\theta \to_z \theta'$. There is none iff θ occurs in $top(\pi)$. The occurrence θ' is Λ-liquid (for a given predicate Λ) iff θ is Λ-liquid.*

This last statement is trivial, because θ and θ' refer to the same occurrence in a term $\rho(t)$ occurring in b as well as in b'.

The following key proposition will be needed in the proofs in Section 4 and 5.

▶ **Proposition 26.** *Let $\theta = (b, \iota, \eta)$ be an occurrence of a variable z in a structured proof (π, ρ) from a TSS P in univariate tytt format, with ι either an incoming occurrence in $top(\pi)$ or the only occurrence of a variable x in s_π. Then $\theta \to_z^* \theta'$ for any occurrence θ' of z in (π, ρ), with * reflexive and transitive closure.*

▶ **Example 27.** In Example 22, the occurrence of z in s_π is $\to_z^*$-related to the three other occurrences of z in the structured proof.

4 Preservation of bounded lookahead

We show that for any TSS P in tyft/tyxt format with bounded lookahead, all P-general rules have bounded lookahead. Thus congruence formats that allow bounded lookahead can be derived by means of the decomposition method from [5].

▶ **Definition 28.** For a proof structure $\pi = (B, \alpha, \varphi)$, let $\prec_\pi$ be the least relation on $var(\pi)$ such that:

- if x occurs in the left-hand side of a premise of π, and y in its right-hand side, then $x \prec_\pi y$, and
- if $b = \frac{H}{t \xrightarrow{a} u} \in B$ and $\varphi(b) = (t' \xrightarrow{a} u')$ with $x \in var(t') \wedge y \in var(t)$ or $x \in var(u) \wedge y \in var(u')$, then $x \prec_\pi y$.

▶ **Observation 29.** *Let $(b, \iota, \eta) \to_z (b', \iota', \eta')$ for two occurrences of a variable z in a proof structure (π, ρ), with ι an occurrence of an $x \in var(\pi)$ in b, and ι' of a $y \in var(\pi)$ in b'. If $b = b'$ then $x = y$, and if $b \neq b'$ then $x \prec_\pi y$.*

▶ **Proposition 30.** *Let $\pi = (B, \alpha, \varphi)$ be a proof structure. If all rules in B have bounded lookahead, then there is no infinite chain $x_0 \prec_\pi x_1 \prec_\pi x_2 \prec_\pi \cdots$.*

▶ **Theorem 31.** *Let P be a TSS in univariate tytt format with bounded lookahead. Then all P-general rules have bounded lookahead.*

Proof. Let P be a TSS with bounded lookahead, and r a P-general rule, say with structured proof (π, ρ). If r had unbounded lookahead, then $top(\pi)$ would contain premises $t_i \xrightarrow{a_i} u_i$ for $i \in \mathbb{Z}_{\geq 0}$ with $var(\rho(u_i)) \cap var(\rho(t_{i+1})) \neq \emptyset$ for all $i \in \mathbb{Z}_{\geq 0}$. Thus, for each $i \in \mathbb{Z}_{\geq 0}$, there would be a $y_i \in var(u_i)$, an $x_{i+1} \in var(t_{i+1})$ and some $z \in var(\rho(y_i)) \cap var(\rho(x_{i+1}))$. Let $\theta_i = (b_i, \iota_i, \eta_i)$ be the occurrence of z in (π, ρ) where b_i is the topmost rule with premise $t_i \xrightarrow{a_i} u_i$, ι_i is the occurrence of y_i in u_i in b_i, and η_i the occurrence of z in $\rho(y_i)$. Likewise,

$\xi_{i+1} = (b_{i+1}, \iota'_{i+1}, \eta'_{i+1})$ is the occurrence of z in (π, ρ) where ι'_{i+1} is the occurrence of x_{i+1} in t_{i+1} in b_{i+1}, and η'_{i+1} the occurrence of z in $\rho(x_{i+1})$. By Proposition 26 $\theta_i \to_z^* \xi_{i+1}$. Now Observation 29 gives $y_i \prec_\pi^* x_{i+1}$. Since by definition also $x_i \prec_\pi y_i$ for all $i \in \mathbb{Z}_{\geq 0}$, we have found an infinite chain $x_0 \prec_\pi x_1 \prec_\pi x_2 \prec_\pi \dots$, which with Proposition 30 yields the required contradiction. ◀

The following example shows that the restriction in Theorem 31 to P-general rules is essential.

▶ **Example 32.** Consider the rule $\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{b} \mathbf{0}}$. By substituting z for both x and y we derive $\frac{z \xrightarrow{a} z}{f(z) \xrightarrow{b} \mathbf{0}}$, which contains unbounded lookahead.

▶ **Corollary 33.** *Theorem 8 applies to each TSS P in tyft/tyxt format with bounded lookahead by choosing for $\mathcal{R}$ the collection of all P-general ruloids.*

5 Preservation of Λ-partial trace safeness

We say that a rule is Λ-*infinitary trace safe* if each Λ-floating variable has at most one occurrence in total in the left-hand sides of the premises and in the target; this occurrence must be at a Λ-liquid position.

▶ **Observation 34.** *Let P be a TSS in univariate tytt format for which each rule is Λ-infinitary trace safe, and (π, ρ) a structured proof of a rule r from P. For each Λ-liquid incoming occurrence θ of z in (π, ρ) there is at most one outgoing occurrence θ' of z with $\theta \to_z \theta'$; this occurrence must be at a Λ-liquid position.*

This holds because by Observation 23, case 2 of Definition 21 applies, with $\theta = (b, \iota, \eta)$ and $\theta = (b, \iota', \eta)$. As θ is Λ-liquid, ι must be a Λ-liquid occurrence of a variable x, and η a Λ-liquid occurrence of z in $\rho(x)$. Since P is in univariate tytt format, ι is Λ-floating. Hence ι' is Λ-liquid.

▶ **Theorem 35.** *Let P be a TSS in univariate tytt format for which each rule is Λ-infinitary trace safe. Then each P-general rule is Λ-infinitary trace safe.*

Proof. Let (π, ρ) be a structured proof from P of a P-general rule r, where $\pi = (B, \alpha, \varphi)$ with $\alpha = (s_\pi \xrightarrow{a} w)$. Let z be a Λ-floating variable of r. Then z has a Λ-liquid occurrence $\theta = (b, \iota, \eta)$ in (π, ρ), with ι either an incoming occurrence in $top(\pi)$ or the only occurrence of a variable x in s_π.

Consider any occurrence of z in the left-hand sides of the premises or the target of r. It corresponds with an occurrence $\theta' = (b', \iota', \eta')$ of z in either a left-hand side of a premise in $top(\pi)$ or the right-hand side of α (thus making ι' the occurrence of w). There is no θ'' with $\theta' \to_z \theta''$. This follows from Observation 25 if ι' occurs in a left-hand side of $top(\pi)$, or from Definition 21 if it is the right-hand side of α.

By Proposition 26, $\theta = \theta_0 \to_z \theta_1 \to_z \cdots \to_z \theta_k = \theta'$. Observations 23, 25 and 34 together imply that any occurrence θ_i of z in this chain is Λ-liquid, and moreover that for each such θ_i the next occurrence θ_{i+1} (if it exists) is uniquely determined. Since moreover $\theta_k \not\to_z$ it follows that θ' is uniquely determined. Thus there is at most one occurrence of z in the left-hand sides of the premises of r or in the target of r, and this occurrence is at a Λ-liquid position. ◀

A tytt rule is Λ-partial trace safe iff it is Λ-infinitary trace safe and has bounded lookahead. Thus, Theorems 31 and 35 together say that if all rules of a TSS P in univariate tytt format are Λ-partial trace safe, then so is each P-general rule.

6 Precongruence of partial trace preorder

To prove that for each TSS in partial trace format the induced partial trace preorder is a precongruence, it suffices to show that each formula in $\mathbb{O}_T$ decomposes into formulas in $\mathbb{O}_T^{\equiv}$.

▶ **Proposition 36.** *Let P be a TSS in partial trace format and $\mathcal{R}$ the set of P-general ruloids. For each term t, $\varphi \in \mathbb{O}_T$, $\psi \in t_{\mathcal{R}}^{-1}(\varphi)$ and variable x:*

$$\psi(x) \equiv \bigwedge_{i \in I} \psi_i \text{ with } \psi_i \in \mathbb{O}_T^{\equiv} \text{ for all } i \in I.$$

▶ **Corollary 37.** *If a TSS is in partial trace format, then the partial trace preorder it induces is a precongruence.*

Proof. Consider a TSS P in partial trace format. Let t be a term and σ, σ' closed substitutions with $\sigma(x) \sqsubseteq_T \sigma'(x)$ for all $x \in var(t)$; we need to prove that $\sigma(t) \sqsubseteq_T \sigma'(t)$. Suppose that $\sigma(t) \models \varphi \in \mathbb{O}_T$. Let $\mathcal{R}$ denote the set of P-general ruloids. By Theorem 8 in combination with Corollary 33 there is a $\psi \in t_{\mathcal{R}}^{-1}(\varphi)$ with $\sigma(x) \models \psi(x)$ for all $x \in var(t)$. By Proposition 36, $\psi(x) \equiv \bigwedge_{i \in I_x} \psi_{i,x}$ with $\psi_{i,x} \in \mathbb{O}_T^{\equiv}$ for all $x \in var(t)$ and $i \in I_x$. So $\sigma(x) \models \psi_{i,x}$ for all $x \in var(t)$ and $i \in I_x$. By Proposition 1, $\mathcal{O}_T^{\equiv}(\sigma(x)) \subseteq \mathcal{O}_T^{\equiv}(\sigma'(x))$ for all $x \in var(t)$. This implies $\sigma'(x) \models \psi_{i,x}$ for all $x \in var(t)$ and $i \in I_x$. So $\sigma'(x) \models \psi(x)$ for all $x \in var(t)$. Therefore, by Theorem 8, $\sigma'(t) \models \varphi$. So $\mathcal{O}_T(\sigma(t)) \subseteq \mathcal{O}_T(\sigma'(t))$. Hence, by Proposition 1, $\sigma(t) \sqsubseteq_T \sigma'(t)$. ◀

7 Conclusion and future work

We introduced the notion of a general rule, which has a proof with minimal variable unifications. To this end we used *proof structures* as alternatives for irredundant proofs, because irredundant proofs abstract away from the variables that are being unified. We moreover showed that if a TSS has bounded lookahead, then the same holds for its general rules. This means that the decomposition method of modal formulas from [13] applies directly to TSSs with bounded lookahead, without first having to turn unbounded into bounded lookahead by means of ordinal count-downs. Both the notion of a general rule and the preservation of bounded lookahead were crucial in the derivation of a congruence format for the partial trace preorder, using the decomposition method.

When restricting attention to TSSs whose rules have finitely many premises, the restriction to bounded lookahead can be dropped from the partial trace format. The reason is that unbounded lookahead is eliminated when converting such a TSS to pure xyft format. With this extension included, our format extends the earlier congruence format for partial trace semantics presented in Bloom [4]. The latter can be seen as the restriction of our format to TSSs in tyft format, allowing only rules with finitely many premises, and requiring Λ to hold universally. The binary Kleene star (see [5]) is an example of an operator that falls in our format, and in that of [5], but not in that of [4]. The application to APC in Section 2.5 falls outside the formats of both [5] and [4].

Future work is to extend the results to TSSs with negative premises, and develop a congruence format for partial trace *equivalence* that allows negative premises. We conjecture that the techniques and results introduced in this paper make it possible to develop congruence formats with lookahead for ready simulation and the decorated trace semantics.

Some applications of lookahead mentioned in the introduction require a richer format, which may be based on the basic format given here. There is a rich body of work extending

existing congruence formats with features like time, probabilities and binders. Recently [8] employed the modal decomposition technique to obtain congruence formats for probabilistic semantics. Our approach lays the groundwork to extend those formats with lookahead.

In [14, 12], modal decomposition is used to derive congruence for weak semantics. By extending this work with lookahead, congruence formats may be developed that e.g. cover the lookahead in the τ-rules from [7]. In [17, Section 8] a congruence format for weak bisimilarity with τ-rules and lookahead is presented, but using a bisimulation-specific method.

Acknowledgement. Paulien de Wind observed that the transformation from unbounded to bounded lookahead in [13] violates the partial trace format.

References

1 P. America and J. W. de Bakker. Designing equivalent semantic models for process creation. *Theoretical Computer Science*, 60(2):109–176, 1988. doi:10.1016/0304-3975(88)90048-5.

2 J. C. M. Baeten and F. W. Vaandrager. An algebra for process creation. *Acta Informatica*, 29(4):303–334, 1992. doi:10.1007/BF01178776.

3 Marco Bernardo. Enriching empa with value passing: A symbolic approach based on lookahead. In *Proc. PAPM 1997*, pages 35–49, 1997.

4 B. Bloom. When is partial trace equivalence adequate? *Formal Aspects of Computing*, 6(3):317–338, 1994. doi:10.1007/BF01215409.

5 B. Bloom, W. J. Fokkink, and R. J. van Glabbeek. Precongruence formats for decorated trace semantics. *Transactions on Computational Logic*, 5(1):26–78, 2004. doi:10.1145/963927.963929.

6 B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995. doi:10.1145/200836.200876.

7 R. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, 1996. doi:10.1145/234752.234756.

8 V. Castiglioni, D. Gebler, and S. Tini. Modal decomposition on nondeterministic probabilistic processes. In *Proc. CONCUR 2016*, volume 59 of *LIPIcs*, pages 36:1–36:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.36.

9 P. R. D'Argenio and M. D. Lee. Probabilistic transition system specification: Congruence and full abstraction of bisimulation. In *Proc. FOSSACS 2012*, volume 7213 of *LNCS*, pages 452–466. Springer, 2012. doi:10.1007/978-3-642-28729-9_30.

10 A. J. Dos Reis. *Compiler Construction Using Java, JavaCC, and Yacc*. Wiley-IEEE, 2012. doi:10.1002/9781118112762.

11 W. J. Fokkink and R. J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation*, 126(1):1–10, 1996. doi:10.1006/inco.1996.0030.

12 W. J. Fokkink and R. J. van Glabbeek. Divide and congruence II: Delay and weak bisimilarity. In *Proc. LICS 2016*, pages 778–787. ACM/IEEE, 2016. doi:10.1145/2933575.2933590.

13 W. J. Fokkink, R. J. van Glabbeek, and P. de Wind. Compositionality of Hennessy-Milner logic by structural operational semantics. *Theoretical Computer Science*, 354(3):421–440, 2006. doi:10.1016/j.tcs.2005.11.035.

14 W. J. Fokkink, R. J. van Glabbeek, and P. de Wind. Divide and congruence: From decomposition of modal formulas to preservation of branching and η-bisimilarity. *Information and Computation*, 214:59–85, 2012. doi:10.1016/j.ic.2011.10.011.

15 R. J. van Glabbeek. Full abstraction in structural operational semantics (extended abstract). In *Proc. AMAST 1993*, Workshops in Computing, pages 75–82. Springer, 1993. doi:10.1007/978-1-4471-3227-1_7.

16 R. J. van Glabbeek. The linear time – branching time spectrum I: The semantics of concrete, sequential processes. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier, 2001.

17 R. J. van Glabbeek. On cool congruence formats for weak bisimulations. *Theoretical Computer Science*, 412(28):3283–3302, 2011. doi:10.1016/j.tcs.2011.02.036.

18 J. F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992. doi:10.1016/0890-5401(92)90013-6.

19 M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985. doi:10.1145/2455.2460.

20 K. V. Hindriks and M. B. van Riemsdijk. Satisfying maintenance goals. In *Proc. DALT 2007*, volume 4897 of *LNCS*, pages 86–103. Springer, 2007. doi:10.1007/978-3-540-77564-5_6.

21 K. G. Larsen. *Context-Dependent Bisimulation between Processes*. PhD thesis, University of Edinburgh, 1986.

22 L. Léonard and G. Leduc. A formal definition of time in LOTOS. *Formal Aspects of Computing*, 10(3):248–266, 1998. doi:10.1007/s001650050015.

23 M. R. Mousavi. Causality in the semantics of Esterel: Revisited. In *Proc. SOS 2009*, volume 18 of *EPTCS*, pages 32–45, 2009. doi:10.4204/EPTCS.18.3.

24 G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004. doi:10.1016/j.jlap.2004.05.001.

25 R. de Simone. Higher-level synchronising devices in Meije–SCCS. *Theoretical Computer Science*, 37(3):245–267, 1985. doi:10.1016/0304-3975(85)90093-3.

A Omitted proofs

A.1 Proof of Proposition 26

Proposition 26 states that for a structured proof (π, ρ), if a variable z occurs in the right-hand side of a premise in $top(\pi)$ or exactly once in s_π, then it is related to all other occurrences of z in π through $\rightarrow_z$. We start with some lemmas needed in the proof of Proposition 26.

▶ **Lemma 38.** *Each variable z occurring in a structured proof (π, ρ) has the form $\rho(x)$ for a prime variable $x \in var(\pi)$.*

Proof. With structural induction on $\rho(y)$ for any $y \in var(\pi)$ we show that each $z \in var(\rho(y))$ has the form $z = \rho(x)$ for a prime variable $x \in var(\pi)$.

- In case y is prime, $\rho(y) \in V$ by the first clause of Definition 17, so $z = \rho(y)$ and we are done.
- Suppose that y is composite. So $y \sim_\pi t$ for some $t \notin V$, and by the third clause of Definition 17, $\rho(y) = \rho(t)$. Hence z occurs in $\rho(t)$, and therefore in $\rho(y')$ for a variable y' occurring in t. Since $\rho(y')$ is a proper subterm of $\rho(y)$, by induction $z = \rho(x)$ for a prime variable $x \in var(\pi)$. ◀

▶ **Lemma 39.** *Let $\theta = \langle b, \iota, \eta\rangle$ and $\theta' = \langle b', \iota', \eta\rangle$ be two occurrences of a variable z in a structured proof (π, ρ) from a TSS P in univariate tytt format, with ι an occurrence of a subterm t in b, and ι' an occurrence of a subterm u in b', where $t \sim_\pi u$. Then $\theta \sim_z \theta'$.*

Proof. We apply induction on the derivation of $t \sim_\pi u$.

- The case that $t \sim_\pi u$ is obtained by the first clause of Definition 14 follows immediate from the definitions, in particular using the first clause of Definition 21, but only when ι and ι' are the indicated occurrences ι_t and $\iota_{t'}$ (or ι_u and $\iota_{u'}$) in Definition 21. We also need to show that if a subterm t occurs multiple times in π, the corresponding occurrences of z in the occurrences of t are related by $\sim_z$. Since t must contain variables, and the sets of variables in different rules $b \in B$ (and α) in a proof structure $\pi = (B, \alpha, \varphi)$ are pairwise disjoint, all occurrences of t lay in the same rule b. The second and third clause of Definition 21 together with the fact that P is in univariate tytt format guarantee that all induced occurrences of z are $\sim_z$-related.
- The case that $t \sim_\pi u$ is obtained by the second clause of Definition 14 is trivial.
- The case that $t \sim_\pi u$ is obtained by reflexivity, symmetry or transitivity is trivial too. ◀

▶ **Lemma 40.** *For each two occurrences θ and θ' of a variable z in a structured proof (π, ρ) from a TSS in univariate tytt format, we have $\theta \sim_z \theta'$.*

Proof. Let $\pi = (B, \alpha, \varphi)$. By Lemma 38, for each variable z occurring in (π, ρ) we can choose an occurrence of the form (b, ι, η) with $b \in B \cup \{\alpha\}$, ι an occurrence of a prime variable x in b, and η the occurrence of z in $\rho(x) = z$. Let (b', ι', η') be another occurrence of z in (π, ρ), with ι' an occurrence of a variable y' in b' and η' an occurrence of z in $\rho(y')$. With structural induction on $\rho(y')$ we show that $(b, \iota, \eta) \sim_z (b', \iota', \eta')$.

- Let y' be prime. Then $\rho(y') \in V$, so $\rho(y') = z = \rho(x)$. By the second clause of Definition 17, $x \sim_\pi y'$. The result now follows from Lemma 39.
- Let y' be composite. Then $y' \sim_\pi t$ for some $t \notin V$, so by the third clause of Definition 17, $\rho(y') = \rho(t)$. Hence the occurrence η' of z in $\rho(y')$ appears as an occurrence η'' of z in $\rho(y'')$ for a variable y'' occurring in t. Let b'' be the rule containing t, ι_t an occurrence of t in b'' and ι'' the appropriate occurrence of y'' within ι_t. Then $(b'', \iota'', \eta'') = \langle b'', \iota_t, \eta' \rangle$ is an occurrence of z in (π, ρ). Since $\rho(y'')$ is a proper subterm of $\rho(y')$, by induction $(b, \iota, \eta) \sim_z (b'', \iota'', \eta'')$. Furthermore, Lemma 39 yields $\langle b'', \iota_t, \eta' \rangle \sim_z \langle b', \iota', \eta' \rangle = (b', \iota', \eta')$. ◀

In the following observations, which are also needed in the proof of Proposition 26, (π, ρ) is a structured proof from a TSS P in univariate tytt format with $\pi = (B, \alpha, \varphi)$.

▶ **Observation 41.** *For each incoming occurrence θ' of z in (π, ρ) there is at most one outgoing occurrence θ of z with $\theta \to_z \theta'$. There is none iff θ' occurs in $top(\pi)$.*

▶ **Observation 42.** *For each outgoing occurrence $\theta' = (b', \iota', \eta')$ of z in (π, ρ) there is at most one incoming occurrence $\theta = (b, \iota, \eta)$ of z with $\theta \to_z \theta'$, where it must be the case that $b' = b \neq \alpha$.*

Now we are ready to present the proof of Proposition 26.

Proof. By Lemma 40, $\theta \sim_z \theta'$. By the definition of $\sim_z$, $\theta = \theta_0 \sim_z^1 \theta_1 \sim_z^1 \cdots \sim_z^1 \theta_k = \theta'$, where $\sim_z^1 = \to_z \cup \leftarrow_z \cup \leftrightsquigarrow_z$. Without loss of generality we assume that there are no repeated occurrences of z in this sequence. By induction on i we show that $\theta_i \to_z \theta_{i+1}$ for all $i = 0, \ldots, k-1$.

- Let $i = 0$, and $\theta_0 = (b, \iota, \eta)$ with ι the only occurrence of x in s_π. By Definition 21 there is no θ_1 with $\theta_1 \to_z \theta_0$ or – using that ι is the only occurrence of x in α – $\theta_0 \leftrightsquigarrow_z \theta_1$.
- Let $i = 0$ and $\theta_0 = (b, \iota, \eta)$ with ι occurring in $top(\pi)$. Since θ_0 is an incoming occurrence, by Definition 21 there is no θ_1 with $\theta_0 \leftrightsquigarrow_z \theta_1$. By Observation 41 there is no θ_1 with $\theta_1 \to_z \theta_0$.

- Let $i > 0$ and θ_i be an incoming occurrence. By Definition 21 there is no θ_{i+1} with $\theta_i \leftrightsquigarrow_z \theta_{i+1}$. By Observation 41 and our convention that $\theta_{i+1} \neq \theta_{i-1}$, we cannot have $\theta_i \leftarrow_z \theta_{i+1}$.
- Let $i > 0$ and θ_i be an outgoing occurrence. By Observation 23 θ_{i-1} must be a incoming occurrence, occurring in the same rule. Hence by Definition 21 there is no θ_{i+1} with $\theta_i \leftrightsquigarrow_z \theta_{i+1}$. By Observation 42 and our convention that $\theta_{i+1} \neq \theta_{i-1}$, we cannot have $\theta_i \leftarrow_z \theta_{i+1}$. ◀

A.2 Proof of Proposition 30

We now present the proof of Proposition 30, which states that if all rules used in a proof structure π have bounded lookahead, then there is no infinite chain $x_0 \prec_\pi x_1 \prec_\pi x_2 \prec_\pi \cdots$.

Proof. With structural induction on proof structures $\pi = (B, \alpha, \varphi)$, seen as well-founded trees. The case that $\alpha \in \mathit{top}(\pi)$ is trivial. So assume $\alpha \notin \mathit{top}(\pi)$.

Let $b_0 \in B$ be the unique rule with $\varphi(b_0) = \alpha$. For each premise β of b_0, let $B_\beta \subseteq B$ be the collection of rules that are above β, and let $\pi_\beta = (B_\beta, \beta, \varphi{\restriction}B_\beta)$. The structured proofs π_β are subproofs of π and by induction do not contain infinite chains as above.

Suppose an infinite chain $x_0 \prec_\pi x_1 \prec_\pi \cdots$ occurred in π. With the possible exception of x_0, which could lay in α, this entire chain can be divided up in connected segments, each of which lays entirely in one of the π_βs. Each segment has at least two variables in it, and two adjacent segments – laying in π_β and π_γ, respectively – overlap in exactly one variable, which must occur in the right-hand side of β as well as in the left-hand side of γ. Here we use that the sets $\mathit{var}(b)$ for $b \in B$ are pairwise disjoint. Using the induction hypothesis, all these segments must be finite. Hence, there must be infinitely many. Restricting the sequence $x_0 \prec_\pi x_1 \prec_\pi x_2 \prec_\pi \cdots$ to those variables that lay in two adjacent segments yields an infinite forward chain of variables in the dependency graph of b_0, contradicting the supposed absence of unbounded lookahead in the rules of B. ◀

A.3 Proof of Proposition 36

Proposition 36 states that given a TSS in partial trace format, the modal decomposition method turns each formula from $\mathbb{O}_T$ into conjunctions of formulas from $\mathbb{O}_T^{\equiv}$.

The following lemma is needed in the proof of Proposition 36. There it is only used in case $x \notin \mathit{var}(t)$, but within the proof of the lemma we also need the case that x has one, Λ-liquid occurrence in t.

▶ **Lemma 43.** *Let P be a TSS in partial trace format, where its rules are Λ-partial trace safe. Let $\mathcal{R}$ denote the set of P-general ruloids. For each term t, $\varphi \in \mathbb{O}_T$, $\psi \in t_{\mathcal{R}}^{-1}(\varphi)$, and variable x that occurs at most once in t, at a Λ-liquid position, we have $\psi(x) \in \mathbb{O}_T^{\equiv}$.*

Proof. We apply induction on the structure of $\varphi \in \mathbb{O}_T$. Let $\psi \in t_{\mathcal{R}}^{-1}(\varphi)$. The two possible syntactic forms of φ in the BNF grammar of $\mathbb{O}_T$ are considered. In case $\varphi = \top$, i.e., $\varphi = \bigwedge_{i \in \emptyset} \varphi_i$, by the second clause of Definition 6, $\psi(x) = \top \in \mathbb{O}_T^{\equiv}$, and we are done. In case $\varphi = \langle a \rangle \varphi'$, by the first clause of Definition 6, $\psi(x) = \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y)[\wedge \chi(x)]$, for some P-general ruloid $r = \frac{H}{t \xrightarrow{a} u}$ and $\chi \in u_{\mathcal{R}}^{-1}(\varphi')$, where the square brackets around the conjunct $\chi(x)$ indicate that it is optional: it is present only if $x \in \mathit{var}(u)$. By Theorems 31 and 35, r is Λ-partial trace safe. Since by assumption x is Λ-floating in r, by Definition 2, x has at most one occurrence in total in the left-hand sides of H and in u; this occurrence must be at a Λ-liquid position. We apply a nested induction on the lookahead of x in H (formally

defined in [13, page 19 – aliased 434]). Suppose first that x occurs in the left-hand side of H, say $x \xrightarrow{c} z$. Since r is tytt, z does not occur in $var(t)$. So by induction on the lookahead of z, $\psi(z) \in \mathbb{O}_T^{\equiv}$. Hence $\psi(x) = \langle c \rangle \psi(z) \in \mathbb{O}_T^{\equiv}$. Suppose now that x does not occur in the left-hand sides of H. Since x occurs at most once in u, at a Λ-liquid position, by induction on the structure of φ', $\chi(x) \in \mathbb{O}_T^{\equiv}$. Hence either $\psi(x) = \chi(x) \in \mathbb{O}_T^{\equiv}$ or $\psi(x) = \top \in \mathbb{O}_T^{\equiv}$. ◀

Now we present the proof of Proposition 36.

Proof. All rules in P are Λ-partial trace safe, for some Λ. We apply induction on the structure of $\varphi \in \mathbb{O}_T$. Let $\psi \in t_{\mathcal{R}}^{-1}(\varphi)$. We consider the two possible syntactic forms of φ in the BNF grammar of $\mathbb{O}_T$. In case $\varphi = \top$, by the second clause of Definition 6, $\psi(x) = \top$, and we are done. In case $\varphi = \langle a \rangle \varphi'$, by the first clause of Definition 6, $\psi(x) = \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y)[\wedge \chi(x)]$, for some P-general ruloid $r = \frac{H}{t \xrightarrow{a} u}$ and $\chi \in u_{\mathcal{R}}^{-1}(\varphi')$, where the conjunct $\chi(x)$ is present only if $x \in var(u)$. By Theorems 31 and 35, r is Λ-partial trace safe. Since r is tytt, for each $(x \xrightarrow{b} y) \in H$, y does not occur in $var(t)$. So by Lemma 43, $\psi(y) \in \mathbb{O}_T^{\equiv}$ for each $(x \xrightarrow{b} y) \in H$. Moreover, by induction, $\chi(x) \equiv \bigwedge_{i \in I} \chi_i$ with $\chi_i \in \mathbb{O}_T^{\equiv}$ for all $i \in I$. Thus $\psi(x)$ is also of this required form. ◀

Capturing Logarithmic Space and Polynomial Time on Chordal Claw-Free Graphs

Berit Grußien

Humboldt-Universität zu Berlin, Berlin, Germany
grussien@informatik.hu-berlin.de

Abstract

We show that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization. $\mathsf{LREC}_=$ is a logic that extends first-order logic with counting by an operator that allows it to formalize a limited form of recursion. This operator can be evaluated in logarithmic space. It follows that there exists a logarithmic-space canonization algorithm for the class of chordal claw-free graphs, and that $\mathsf{LREC}_=$ captures logarithmic space on this graph class. Since $\mathsf{LREC}_=$ is contained in fixed-point logic with counting, we also obtain that fixed-point logic with counting captures polynomial time on the class of chordal claw-free graphs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.4.1 Mathematical Logics

Keywords and phrases Descriptive complexity, logarithmic space, polynomial time, chordal claw-free graphs

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.26

1 Introduction

Descriptive complexity is a field of computational complexity theory that provides logical characterizations for the standard complexity classes. The starting point of descriptive complexity was a theorem of Fagin in 1974 [6], which states that existential second-order logic characterizes, or *captures*, the complexity class NP. Later, similar logical characterizations were found for further complexity classes. For example, Immerman proved that deterministic transitive closure logic DTC captures LOGSPACE [19], and independently of one another, Immerman [18] and Vardi [24] showed that fixed-point logic FP captures PTIME[1]. However, these two results have a draw-back: They only hold on ordered structures, that is, on structures with a distinguished binary relation which is a linear order on the universe of the structure. On structures that are not necessarily ordered, there have only been partial results towards capturing LOGSPACE or PTIME, so far.

A negative partial result towards capturing LOGSPACE follows from Etessami and Immerman's result that (directed) tree isomorphism is not definable in transitive closure logic with counting TC+C [5]. This implies that tree isomorphism is neither definable in deterministic nor in symmetric transitive closure logic with counting (DTC+C and STC+C), although it is decidable in LOGSPACE [22]. Hence, DTC+C and STC+C are not strong enough to capture LOGSPACE even on the class of trees. That is why, in 2011 a new logic with logarithmic-space data complexity was introduced [13, 14]. This logic, $\mathsf{LREC}_=$, is an extension of first-order logic with counting by an operator that allows a limited form of

[1] More precisely, Immerman and Vardi's theorem holds for least fixed-point logic (LFP) and the equally expressive inflationary fixed-point logic (IFP). Our indeterminate FP refers to either of these two logics.

recursion. $\mathsf{LREC}_=$ strictly contains $\mathsf{STC}+\mathsf{C}$ and $\mathsf{DTC}+\mathsf{C}$. In [13, 14], the authors proved that $\mathsf{LREC}_=$ captures $\mathsf{LOGSPACE}$ on the class of (directed) trees and on the class of interval graphs. In this paper we now show that $\mathsf{LREC}_=$ captures $\mathsf{LOGSPACE}$ also on the class of chordal claw-free graphs. More precisely, this paper's technical main contribution states that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization. This does not only imply that $\mathsf{LREC}_=$ captures $\mathsf{LOGSPACE}$ on chordal claw-free graphs, but also that there exists a logarithmic-space canonization algorithm for the class of chordal claw-free graphs. Hence, the isomorphism and automorphism problem for this graph class is solvable in logarithmic space.

For polynomial time there also exist partial characterizations. Fixed-point logic with counting $\mathsf{FP}+\mathsf{C}$ captures PTIME, for example, on planar graphs [8], on all classes of graphs of bounded treewidth [15] and on K_5-minor free graphs [9]. Note that all these classes can be defined by a list of forbidden minors. In fact, Grohe showed in 2010 that $\mathsf{FP}+\mathsf{C}$ captures PTIME on all graph classes with excluded minors [11]. Instead of graph classes with excluded minors, one can also consider graph classes with excluded induced subgraphs, i.e. graph classes $\mathcal{C}$ that are closed under taking induced subgraphs. For some of these graph classes $\mathcal{C}$, e.g. chordal graphs [10], comparability graphs [21] and co-comparability graphs [21], capturing PTIME on $\mathcal{C}$ is as hard as capturing PTIME on the class of all graphs for any "reasonable" logic.[2] This gives us reason to consider subclasses of chordal graphs, comparability graphs and co-comparability graphs more closely. There are results showing that $\mathsf{FP}+\mathsf{C}$ captures PTIME on interval graphs (chordal co-comparability graphs) [20], on permutation graphs (comparability co-comparability graphs) [17] and on chordal comparability graphs [16]. Further, Grohe proved that $\mathsf{FP}+\mathsf{C}$ captures PTIME on chordal line graphs [10]. At the same time he conjectured that this is also the case for the class of chordal claw-free graphs, which is an extension of the class of chordal line graphs. Our main result implies that Grohe's conjecture is true: Since $\mathsf{LREC}_=$ is contained in $\mathsf{FP}+\mathsf{C}$, it yields that there exists an $\mathsf{FP}+\mathsf{C}$-canonization of the class of chordal claw-free graphs. Hence, $\mathsf{FP}+\mathsf{C}$ captures PTIME also on the class of chordal claw-free graphs.

Our main result is based on a study of chordal claw-free graphs. Chordal graphs are the intersection graphs of subtrees of a tree [2, 7, 25], and a clique tree of a chordal graph corresponds to a minimal representation of the graph as such an intersection graph. We prove that chordal claw-free graphs are (claw-free) intersection graphs of paths in a tree, and that for each connected chordal claw-free graph the clique tree is unique.

1.1 Structure

The preliminaries in Section 2 will be followed by a Section 3 where we analyze the structure of clique trees of chordal claw-free graphs, and, e.g., show that connected chordal claw-free graphs have a unique clique tree. In Section 4, we transform the clique tree of a connected chordal claw-free graph into a directed tree, and color each maximal clique with information about its intersection with other maximal cliques by using a special coloring with a linearly ordered set of colors. We obtain what we call the supplemented clique tree, and show that it is definable in $\mathsf{STC}+\mathsf{C}$ by means of a parameterized transduction. We know that there exists an $\mathsf{LREC}_=$-canonization of colored trees if the set of colors is linearly ordered [13, 14]. We apply this $\mathsf{LREC}_=$-canonization to the supplemented clique tree in Section 5 and obtain the

[2] Note that $\mathsf{FP}+\mathsf{C}$ does not capture PTIME on the class of all graphs [3]. Hence, it does not capture PTIME on the class of chordal graphs, comparability graphs or co-comparability graphs either.

canon of this colored directed tree. Due to the type of coloring, the information about the maximal cliques is also contained in the colors of the canon of the supplemented clique tree. This information and the linear order on the vertices of the canon of the supplemented clique tree allow us to define the maximal cliques of a canon of the connected chordal claw-free graph, from which we can easily construct the canon of the graph. By combining the canons of the connected components, we obtain a canon for each chordal claw-free graph.

2 Basic Definitions and Notation

We write $\mathbb{N}$ for the set of all non-negative integers. For all $n, n' \in \mathbb{N}$, we define $[n, n'] := \{m \in \mathbb{N} \mid n \leq m \leq n'\}$ and $[n] := [1, n]$. We often denote tuples $(a_1, \ldots, a_k)$ by $\bar{a}$. Given a tuple $\bar{a} = (a_1, \ldots, a_k)$, let $\tilde{a} := \{a_1, \ldots, a_k\}$. Let $n \geq 1$. Let $\bar{a}^i = (a^i_1, \ldots, a^i_{k_i})$ be a tuple of length k_i for each $i \in [n]$. We denote the tuple $(a^1_1, \ldots, a^1_{k_1}, \ldots, a^n_1, \ldots, a^n_{k_n})$ by $(\bar{a}^1, \ldots, \bar{a}^n)$. Mappings $f\colon A \to B$ are extended to tuples $\bar{a} = (a_1, \ldots, a_k)$ over A via $f(\bar{a}) := (f(a_1), \ldots, f(a_k))$. Let $\approx$ be an equivalence relation on a set S. Then $a/_\approx$ denotes the equivalence class of $a \in S$ with respect to $\approx$. For $\bar{a} = (a_1, \ldots, a_n) \in S^n$ and $R \subseteq S^n$, we let $\bar{a}/_\approx := (a_1/_\approx, \ldots, a_n/_\approx)$ and $R/_\approx := \{\bar{a}/_\approx \mid \bar{a} \in R\}$. A *partition* of a set S is a set $\mathcal{P}$ of disjoint non-empty subsets of S where $S = \bigcup_{A \in \mathcal{P}} A$. For a set S, we let $\binom{S}{2}$ be the set of all 2-element subsets of S.

2.1 Graphs and LO-colored Graphs

A *graph* is a pair (V, E) consisting of a non-empty finite set V of *vertices* and a set $E \subseteq \binom{V}{2}$ of *edges*. Let $G = (V, E)$ and $G' = (V', E')$ be graphs. The *union* $G \cup G'$ of G and G' is the graph $(V \cup V', E \cup E')$. For a subset $W \subseteq V$ of vertices, $G[W]$ denotes the *induced subgraph* of G with vertex set W. *Connectivity* and *connected components* are defined in the usual way. We denote the *neighbors* of a vertex $v \in V$ by $N(v)$. A set $B \subseteq V$ is a *clique* if $\binom{B}{2} \subseteq E$. A maximal clique, or *max clique*, is a clique that is not properly contained in any other clique.

A graph is *chordal* if all its cycles of length at least 4 have a chord, which is an edge that connects two non-consecutive vertices of the cycle. A *claw-free* graph is a graph that does not have a *claw*, i.e. a graph isomorphic to the complete bipartite graph $K_{1,3}$, as an induced subgraph. We denote the class of (connected) chordal claw-free graphs by (con-)CCF.

A subgraph P of G is a *path* of G if $P = (\{v_0, \ldots, v_k\}, \{\{v_0, v_1\}, \ldots, \{v_{k-1}, v_k\}\})$ for distinct vertices $v_0, \ldots, v_k$ of G. We also denote path P by the sequence $v_0, \ldots, v_k$ of vertices. We let v_0 and v_k be the *ends* of P. A connected acyclic graph is a *tree*. Let $T = (V, E)$ be a tree. A *subtree* of T is a connected subgraph of T. A vertex $v \in V$ of degree 1 is called a *leaf*.

A pair (V, E) is a *directed graph* if V is a non-empty finite set and $E \subseteq V^2$. A connected acyclic directed graph where the in-degree of each vertex is at most 1 is a *directed tree*. Let $T = (V, E)$ be a directed tree. The vertex of in-degree 0 is the *root* of T. If $(v, w) \in E$, then w is a *child* of v, and v the *parent* of w. Let w, w' be children of $v \in V$. Then w is a *sibling* of w' if $w \neq w'$. If there is a (directed) path from $v \in V$ to $w \in V$ in T, then v is an *ancestor* of w.

Let $G = (V, E)$ be a graph and $f\colon V \to C$ be a mapping from the vertices of G to a finite set C. Then f is a *coloring* of G, and the elements of C are called *colors*. In this paper we color the vertices of a graph with binary relations on a linearly ordered set. We call graphs with such a coloring LO-*colored graphs*. More precisely, an LO-colored graph is a tuple $G = (V, E, M, \trianglelefteq, L)$ with the following four properties:

1. The pair (V, E) is a graph. We call (V, E) the *underlying graph* of G.
2. The set of *basic color elements* M is a non-empty finite set with $M \cap V = \emptyset$.
3. The binary relation $\trianglelefteq \subseteq M^2$ is a linear order on M.
4. The relation $L \subseteq V \times M^2$ assigns to every $v \in V$ an LO-*color* $L_v := \{(d, d') \mid (v, d, d') \in L\}$.

Let $d_0, \dots, d_{|M|-1}$ be the enumeration of the basic color elements in M according to their linear order $\trianglelefteq$. We call $L_v^{\mathbb{N}} := \{(i,j) \in \mathbb{N}^2 \mid (d_i, d_j) \in L_v\}$ the NO-*color* of $v \in V$.

We can use the linear order $\trianglelefteq$ on M to obtain a linear order on the colors $\{L_v \mid v \in V\}$ of G. Thus, an LO-colored graph is a special kind of colored graph with a linear order on its colors.

2.2 Structures

A *vocabulary* is a finite set τ of relation symbols. Each relation symbol $R \in \tau$ has a fixed arity $\mathrm{ar}(R) \in \mathbb{N}$. A *$\tau$-structure* consists of a non-empty finite set $U(A)$, its *universe*, and for each relation symbol $R \in \tau$ of a relation $R(A) \subseteq U(A)^{\mathrm{ar}(R)}$.

An *isomorphism* between τ-structures A and B is a bijection $f\colon U(A) \to U(B)$ such that for all $R \in \tau$ and all $\bar{a} \in U(A)^{\mathrm{ar}(R)}$ we have $\bar{a} \in R(A)$ if and only if $f(\bar{a}) \in R(B)$. We write $A \cong B$ to indicate that A and B are *isomorphic*.

Let E be a binary relation symbol. Each graph corresponds to an $\{E\}$-structure $G = (V, E)$ where the universe V is the vertex set and E is an irreflexive and symmetric binary relation, the edge relation. To represent an LO-colored graph $G = (V, E, M, \trianglelefteq, L)$ as a logical structure we extend the 5-tuple by a set U to a 6-tuple $(U, V, E, M, \trianglelefteq, L)$, and we require that $U = V \dot\cup M$ in addition to the properties 1-4. The set U serves as the universe of the structure, and $V, E, M, \trianglelefteq, L$ are relations on U. We usually do not distinguish between (LO-colored) graphs and their representation as logical structures. It will be clear from the context which form we are referring to.

2.3 Logics

In this section we introduce symmetric transitive closure logic (with counting) and $\mathsf{LREC}_=$.

We assume basic knowledge in logic, in particular of *first-order logic* (FO). *First-order logic with counting* (FO+C) extends FO by a counting operator that allows for counting the cardinality of FO+C-definable relations. It lives in a two-sorted context, where structures A are equipped with a *number sort* $N(A) := [0, |U(A)|]$. FO+C has two types of variables: FO+C-variables are either *structure variables* that range over the universe $U(A)$ of a structure A, or *number variables* that range over the number sort $N(A)$. For each variable u, let $A^u := U(A)$ if u is a structure variable, and $A^u := N(A)$ if u is a number variable. Let $A^{(u_1,\dots,u_k)} := A^{u_1} \times \cdots \times A^{u_k}$. Tuples $(u_1, \dots, u_k)$ and $(v_1, \dots, v_\ell)$ of variables are *compatible* if $k = \ell$, and for every $i \in [k]$ the variables u_i and v_i have the same type. An *assignment in A* is a mapping α from the set of variables to $U(A) \cup N(A)$, where for each variable u we have $\alpha(u) \in A^u$. For tuples $\bar{u} = (u_1, \dots, u_k)$ of variables and $\bar{a} = (a_1, \dots, a_k) \in A^{\bar{u}}$, the assignment $\alpha[\bar{a}/\bar{u}]$ maps u_i to a_i for each $i \in [k]$, and each variable $v \notin \tilde{u}$ to $\alpha(v)$. By $\varphi(u_1, \dots, u_k)$ we denote a formula φ with $\mathrm{free}(\varphi) \subseteq \{u_1, \dots, u_k\}$, where $\mathrm{free}(\varphi)$ is the set of free variables in φ. Given a formula $\varphi(u_1, \dots, u_k)$, a structure A and $(a_1, \dots, a_k) \in A^{(u_1,\dots,u_k)}$, we write $A \models \varphi[a_1, \dots, a_k]$ if φ holds in A with u_i assigned to a_i for each $i \in [k]$. We write $\varphi[A, \alpha; \bar{u}]$ for the set of all tuples $\bar{a} \in A^{\bar{u}}$ with $(A, \alpha[\bar{a}/\bar{u}]) \models \varphi$. For a formula $\varphi(\bar{u})$ (with $\mathrm{free}(\varphi) \subseteq \tilde{u}$) we also denote $\varphi[A, \alpha; \bar{u}]$ by $\varphi[A; \bar{u}]$, and for a formula $\varphi(\bar{v}, \bar{u})$ and $\bar{a} \in A^{\bar{v}}$, we denote $\varphi[A, \alpha[\bar{a}/\bar{v}]; \bar{u}]$ also by $\varphi[A, \bar{a}; \bar{u}]$.

FO+C is obtained by extending FO with the following formula formation rules:

- $\phi := p \le q$ is a formula if p, q are number variables. We let $\mathrm{free}(\phi) := \{p, q\}$.
- $\phi' := \#\bar{u}\, \psi = \bar{p}$ is a formula if ψ is a formula, $\bar{u}$ is a tuple of variables and $\bar{p}$ a tuple of number variables. We let $\mathrm{free}(\phi') := (\mathrm{free}(\psi) \setminus \tilde{u}) \cup \tilde{p}$.

To define the semantics, let A be a structure and α be an assignment. We let

- $(A, \alpha) \models p \le q$ iff $\alpha(p) \le \alpha(q)$,
- $(A, \alpha) \models \#\bar{u}\, \psi = \bar{p}$ iff $|\psi[A, \alpha; \bar{u}]| = \langle \alpha(\bar{p}) \rangle_A$,

where for tuples $\bar{n} = (n_1, \dots, n_k) \in N(A)^k$ we let $\langle \bar{n} \rangle_A$ be the number

$$\langle \bar{n} \rangle_A \;:=\; \sum_{i=1}^{k} n_i \cdot (|U(A)| + 1)^{i-1}.$$

Symmetric transitive closure logic (with counting) $\mathsf{STC}(+\mathsf{C})$ is an extension of $\mathsf{FO}(+\mathsf{C})$ with stc-operators. The set of all $\mathsf{STC}(+\mathsf{C})$-formulas is obtained by extending the formula formation rules of $\mathsf{FO}(+\mathsf{C})$ by the following rule:

- $\phi := [\mathrm{stc}_{\bar{u},\bar{v}}\, \psi](\bar{u}', \bar{v}')$ is a formula if ψ is a formula and $\bar{u}, \bar{v}, \bar{u}', \bar{v}'$ are compatible tuples of structure (and number) variables. We let $\mathrm{free}(\phi) := \tilde{u}' \cup \tilde{v}' \cup \big(\mathrm{free}(\psi) \setminus (\tilde{u} \cup \tilde{v})\big)$.

Let A be a structure and α be an assignment. We let

- $(A, \alpha) \models [\mathrm{stc}_{\bar{u},\bar{v}}\, \psi](\bar{u}', \bar{v}')$ iff $(\alpha(\bar{u}'), \alpha(\bar{v}'))$ is contained in the symmetric transitive closure of $\psi[A, \alpha; \bar{u}, \bar{v}]$.

$\mathsf{LREC}_=$ is an extension of $\mathsf{FO}{+}\mathsf{C}$ with lrec-operators, which allow a limited form of recursion. We extend the formula formation rules of $\mathsf{FO}{+}\mathsf{C}$ by the following rule:

- $\phi := [\mathrm{lrec}_{\bar{u},\bar{v},\bar{p}}\ \varphi_=,\ \varphi_{\mathtt{E}},\ \varphi_{\mathtt{C}}](\bar{w}, \bar{r})$ is a formula if $\varphi_=$, $\varphi_{\mathtt{E}}$ and $\varphi_{\mathtt{C}}$ are formulas, $\bar{u}, \bar{v}, \bar{w}$ are compatible tuples of variables and $\bar{p}, \bar{r}$ are non-empty tuples of number variables.

  We let $\mathrm{free}(\phi) := \big(\mathrm{free}(\varphi_=) \setminus (\tilde{u} \cup \tilde{v})\big) \cup \big(\mathrm{free}(\varphi_{\mathtt{E}}) \setminus (\tilde{u} \cup \tilde{v})\big) \cup \big(\mathrm{free}(\varphi_{\mathtt{C}}) \setminus (\tilde{u} \cup \tilde{p})\big) \cup \tilde{w} \cup \tilde{r}$.

Let A be a structure and α be an assignment. We let

- $(A, \alpha) \models [\mathrm{lrec}_{\bar{u},\bar{v},\bar{p}}\ \varphi_=,\ \varphi_{\mathtt{E}},\ \varphi_{\mathtt{C}}](\bar{w}, \bar{r})$ iff $\big(\alpha(\bar{w})/_\sim, \langle \alpha(\bar{r}) \rangle_A\big) \in X$,

where X and $\sim$ are defined as follows: Let $\mathtt{V}_0 := A^{\bar{u}}$ and $\mathtt{E}_0 := \varphi_{\mathtt{E}}[A, \alpha; \bar{u}, \bar{v}]$. We define $\sim$ to be the reflexive, symmetric, transitive closure of the binary relation $\varphi_=[A, \alpha; \bar{u}, \bar{v}]$ over $\mathtt{V}_0$. Now consider the graph $\mathtt{G} = (\mathtt{V}, \mathtt{E})$ with $\mathtt{V} := \mathtt{V}_0/_\sim$ and $\mathtt{E} := \{(\bar{a}/_\sim, \bar{b}/_\sim) \in \mathtt{V}^2 \mid (\bar{a}, \bar{b}) \in \mathtt{E}_0\}$. To every $\bar{a}/_\sim \in \mathtt{V}$ we assign the set $\mathtt{C}(\bar{a}/_\sim) := \{\langle \bar{n} \rangle_A \mid \text{there is an } \bar{a}' \in \bar{a}/_\sim \text{ with } \bar{n} \in \varphi_{\mathtt{C}}[A, \alpha[\bar{a}'/\bar{u}]; \bar{p}]\}$ of numbers. Let $\bar{a}/_\sim \mathtt{E} := \{\bar{b}/_\sim \in \mathtt{V} \mid (\bar{a}/_\sim, \bar{b}/_\sim) \in \mathtt{E}\}$ and $\mathtt{E}\bar{b}/_\sim := \{\bar{a}/_\sim \in \mathtt{V} \mid (\bar{a}/_\sim, \bar{b}/_\sim) \in \mathtt{E}\}$. Then, for all $\bar{a}/_\sim \in \mathtt{V}$ and $\ell \in \mathbb{N}$,

$$(\bar{a}/_\sim, \ell) \in X \;:\Longleftrightarrow\; \ell > 0 \text{ and } \left|\left\{\bar{b}/_\sim \in \bar{a}/_\sim \mathtt{E} \;\middle|\; \left(\bar{b}/_\sim, \left\lfloor \frac{\ell - 1}{|\mathtt{E}\bar{b}/_\sim|} \right\rfloor\right) \in X\right\}\right| \in \mathtt{C}(\bar{a}/_\sim).$$

$\mathsf{LREC}_=$ semantically contains $\mathsf{STC}{+}\mathsf{C}$ [14]. Note that simple arithmetics like addition and multiplication are definable in $\mathsf{STC}{+}\mathsf{C}$, and therefore, in $\mathsf{LREC}_=$.

2.4 Transductions

Transductions (also known as *syntactical interpretations*) define certain structures within other structures. More on transductions can be found in [12, 16]. In the following we introduce parameterized transductions for $\mathsf{FO}(+\mathsf{C})$, $\mathsf{STC}(+\mathsf{C})$ and $\mathsf{LREC}_=$.

▶ **Definition 2.1** (Parameterized Transduction)**.** Let τ_1, τ_2 be vocabularies, and let L be a logic that extends FO.

1. A *parameterized* $\mathsf{L}[\tau_1, \tau_2]$*-transduction* is a tuple

$$\Theta(\bar{x}) \;=\; \Big(\theta_{\mathrm{dom}}(\bar{x}), \theta_U(\bar{x}, \bar{u}), \theta_{\approx}(\bar{x}, \bar{u}, \bar{u}'), \big(\theta_R(\bar{x}, \bar{u}_{R,1}, \dots, \bar{u}_{R,\mathrm{ar}(R)})\big)_{R \in \tau_2}\Big)$$

   of $\mathsf{L}[\tau_1]$-formulas, where $\bar{x}$ is a tuple of structure variables, and $\bar{u}, \bar{u}'$ and $\bar{u}_{R,i}$ for every $R \in \tau_2$ and $i \in [\mathrm{ar}(R)]$ are compatible tuples of variables.
2. The *domain* of $\Theta(\bar{x})$ is the class $\mathrm{Dom}(\Theta(\bar{x}))$ of all pairs $(A, \bar{p})$ such that $A \models \theta_{\mathrm{dom}}[\bar{p}]$, $\theta_U[A, \bar{p}; \bar{u}]$ is not empty and $\theta_{\approx}[A, \bar{p}; \bar{u}, \bar{u}']$ is an equivalence relation, where A is a τ_1-structure and $\bar{p} \in A^{\bar{x}}$. The elements in $\bar{p}$ are called *parameters*.

3. Let $(A, \bar{p})$ be in the domain of $\Theta(\bar{x})$, and let us denote $\theta_{\approx}[A, \bar{p}; \bar{u}, \bar{u}']$ by $\approx$. We define a τ_2-structure $\Theta[A, \bar{p}]$ as follows. We let

$$U(\Theta[A, \bar{p}]) := \theta_U[A, \bar{p}; \bar{u}]/_{\approx}$$

be the universe of $\Theta[A, \bar{p}]$. Further, for each $R \in \tau_2$, we let

$$R(\Theta[A, \bar{p}]) := \left(\theta_R[A, \bar{p}; \bar{u}_{R,1}, \ldots, \bar{u}_{R,\mathrm{ar}(R)}] \cap \theta_U[A, \bar{p}; \bar{u}]^{\mathrm{ar}(R)}\right)\Big/_{\approx}.$$

A parameterized $\mathsf{L}[\tau_1, \tau_2]$-transduction defines a parameterized mapping from τ_1-structures into τ_2-structures via $\mathsf{L}[\tau_1]$-formulas. A parameterized $\mathsf{L}[\tau_1, \tau_2]$-transduction $\Theta(\bar{x})$ is an $\mathsf{L}[\tau_1, \tau_2]$-*transduction* if $\bar{x}$ is the empty tuple. If $\theta_{\mathrm{dom}} := \top$ or $\theta_{\approx} := \bot$, we omit the respective formula in the presentation of the transduction.

An important property of $\mathsf{L}[\tau_1, \tau_2]$-transductions is that, for suitable logics L, they allow to *pull back* $\mathsf{L}[\tau_2]$-formulas, which means that for each $\mathsf{L}[\tau_2]$-formula there exists an $\mathsf{L}[\tau_1]$-formula that expresses essentially the same. Logic L is *closed under (parameterized)* L*-transductions* if for all vocabularies τ_1, τ_2 each (parameterized) $\mathsf{L}[\tau_1, \tau_2]$-transduction allows to pull back $\mathsf{L}[\tau_2]$-formulas. Each logic $\mathsf{L} \in \{\mathsf{FO}(+\mathsf{C}), \mathsf{STC}(+\mathsf{C}), \mathsf{LREC}_=\}$ is closed under (parameterized) $\mathsf{L}[\tau_1, \tau_2]$-transductions [4, 14, 16].

2.5 Canonization

In this section we introduce ordered structures, (definable) canonization and the capturing of the complexity class $\mathsf{LOGSPACE}$.

Let τ be a vocabulary with $\leq \notin \tau$. A $\tau \cup \{\leq\}$-structure A' is *ordered* if the relation symbol $\leq$ is interpreted as a linear order on the universe of A'. Let A be a τ-structure. A $\tau \cup \{\leq\}$-structure A' is an *ordered copy* of A if $A'|_\tau \cong A$. Let $\mathcal{C}$ be a class of τ-structures. A mapping f is a *canonization mapping* of $\mathcal{C}$ if it assigns every structure $A \in \mathcal{C}$ to an ordered copy $f(A) = (A_f, \leq_f)$ of A such that for all structures $A, B \in \mathcal{C}$ we have $f(A) \cong f(B)$ if $A \cong B$. We call the ordered structure $f(A)$ the *canon* of A.

Let L be a logic that extends FO. Let $\Theta(\bar{x})$ be a parameterized $\mathsf{L}[\tau, \tau \cup \{\leq\}]$-transduction, where $\bar{x}$ is a tuple of structure variables. We say $\Theta(\bar{x})$ *canonizes* a τ-structure A if there exists a tuple $\bar{p} \in A^{\bar{x}}$ such that $(A, \bar{p}) \in \mathrm{Dom}(\Theta(\bar{x}))$, and for all tuples $\bar{p} \in A^{\bar{x}}$ with $(A, \bar{p}) \in \mathrm{Dom}(\Theta(\bar{x}))$, the $\tau \cup \{\leq\}$-structure $\Theta[A, \bar{p}]$ is an ordered copy of A.[3] A *(parameterized)* L*-canonization* of a class $\mathcal{C}$ of τ-structures is a (parameterized) $\mathsf{L}[\tau, \tau \cup \{\leq\}]$-transduction that canonizes all $A \in \mathcal{C}$. A class $\mathcal{C}$ of τ-structures *admits* L*-definable canonization* if $\mathcal{C}$ has a (parameterized) L-canonization.

The following proposition and theorem are essential for proving that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization in Section 5.

▶ **Proposition 2.2** ([12][4]). *Let $\mathcal{C}$ be a class of graphs, and $\mathcal{C}_{conn}$ be the class of all connected components of the graphs in $\mathcal{C}$. If $\mathcal{C}_{conn}$ admits $\mathsf{LREC}_=$-definable canonization, then $\mathcal{C}$ does as well.*

[3] Note that if the tuple $\bar{x}$ of parameter variables is the empty tuple, $\mathsf{L}[\tau, \tau \cup \{\leq\}]$-transduction Θ canonizes a τ-structure A if $A \in \mathrm{Dom}(\Theta)$ and the $\tau \cup \{\leq\}$-structure $\Theta[A]$ is an ordered copy of A.

[4] In [12, Corollary 3.3.21] Proposition 2.2 is only shown for $\mathsf{IFP}+\mathsf{C}$. The proof of Corollary 3.3.21 uses Lemma 3.3.18, the Transduction Lemma, and that connectivity and simple arithmetics are definable. As $\mathsf{LREC}_=$ is closed under parameterized $\mathsf{LREC}_=$-transductions, the Transduction Lemma also holds for $\mathsf{LREC}_=$ [14]. Connectivity and all arithmetics (e.g. addition, multiplication and Fact 3.3.14) that are necessary to show Lemma 3.3.18 and Corollary 3.3.21 can also be defined in $\mathsf{LREC}_=$. Further, Lemma 3.3.12 and 3.3.17, which are used to prove Lemma 3.3.18 can be shown by pulling back simple FO-formulas under $\mathsf{LREC}_=$-transductions. Hence, Corollary 3.3.21 also holds for $\mathsf{LREC}_=$.

▶ **Theorem 2.3** ([14, 16][5]). *The class of* LO*-colored directed trees admits* $\mathsf{LREC}_=$*-definable canonization.*

We can use definable canonization of a graph class to prove that LOGSPACE is captured on this graph class. Let L be a logic and $\mathcal{C}$ be a graph class. L *captures* LOGSPACE *on* $\mathcal{C}$ if for each class $\mathcal{D} \subseteq \mathcal{C}$, there exists an L-sentence defining $\mathcal{D}$ if and only if $\mathcal{D}$ is LOGSPACE-decidable.[6] A fundamental result was shown by Immerman:[7]

▶ **Theorem 2.4** ([19]). DTC *captures* LOGSPACE *on the class of all ordered graphs.*

Deterministic transitive closure logic DTC is a logic that is contained in $\mathsf{LREC}_=$ [14]. Therefore, we obtain the following corollary:

▶ **Corollary 2.5.** $\mathsf{LREC}_=$ *captures* LOGSPACE *on the class of all ordered graphs.*

Let us suppose there exists a parameterized $\mathsf{LREC}_=$-canonization of a graph class $\mathcal{C}$. Since $\mathsf{LREC}_=$ captures LOGSPACE on the class of all ordered graphs and we can pull back each $\mathsf{LREC}_=$-sentence that defines a logarithmic-space property on ordered graphs under this canonization, the capturing result transfers from ordered graphs to the class $\mathcal{C}$.

▶ **Proposition 2.6.** *Let* $\mathcal{C}$ *be a class of graphs. If* $\mathcal{C}$ *admits* $\mathsf{LREC}_=$*-definable canonization, then* $\mathsf{LREC}_=$ *captures* LOGSPACE *on* $\mathcal{C}$*.*

3 Structure of Clique Trees

Clique trees of connected chordal claw-free graphs play an important role in the subsequent canonization of chordal claw-free graphs. Thus, we analyze the structure of clique trees of connected chordal claw-free graphs in this section.

First we introduce clique trees of chordal graphs. Then we show that chordal claw-free graphs are intersection graphs of paths of a tree. We use this property to prove that each connected chordal claw-free graph has a unique clique tree. Finally, we introduce two different types of max cliques in a clique tree, star cliques and fork cliques, and show that each max clique of a connected chordal claw-free graph is of one of these types if its degree in the clique tree is at least 3.

Chordal graphs are precisely the intersection graphs of subtrees of a tree. A clique tree of a chordal graph G specifies a minimal representation of G as an intersection graph of subtrees of a tree. Clique trees were introduced independently by Buneman [2], Gavril [7] and Walter [25]. A detailed introduction of chordal graphs and their clique trees can be found in [1].

Let G be a chordal graph, and let $\mathcal{M}$ be the set of max cliques of G. Further, let $\mathcal{M}_v$ be the set of all max cliques in $\mathcal{M}$ that contain a vertex v of G. A *clique tree* of G is a tree $T = (\mathcal{M}, \mathcal{E})$ whose vertex set is the set $\mathcal{M}$ of all max cliques, where for all $v \in V$ the

[5] It is shown in [14, Remark 4.8], and in more detail in [16, Section 8.4] that the class of all colored directed trees that have a linear order on the colors admits LREC-definable canonization. This can easily be extended to LO-colored directed trees since an LO-colored graph is a special kind of colored graph that has a linear order on its colors. LREC is contained in $\mathsf{LREC}_=$ [14].

[6] A precise definition of what it means that a logic *(effectively strongly) captures* a complexity class can be found in [4, Chapter 11].

[7] Immerman proved this capturing result not only for the class of ordered graphs but for the class of ordered structures.

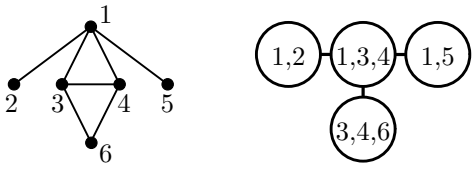


Figure 1 A chordal graph and a clique tree of the graph.

induced subgraph $T[\mathcal{M}_v]$ is connected. Hence, for each $v \in V$ the induced subgraph $T[\mathcal{M}_v]$ is a subtree of T. Then G is the intersection graph of the subtrees $T[\mathcal{M}_v]$ of T where $v \in V$. An example of a clique tree of a chordal graph is shown in Figure 1.

Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal graph G. It is easy to see that clique tree T satisfies the *clique-intersection property*: Let $M_1, M_2, M_3 \in \mathcal{M}$ be vertices of the tree T. If M_2 is on the path from M_1 to M_3, then $M_1 \cap M_3 \subseteq M_2$.

In the following we consider the class CCF of chordal claw-free graphs. For each vertex v of a chordal claw-free graph, we prove that the set of max cliques $\mathcal{M}_v$ induces a path in each clique tree. Consequently, chordal claw-free graphs are intersection graphs of paths of a tree. Note that not all intersection graphs of paths of a tree are claw-free (see Figure 1).

▶ **Lemma 3.1.** *Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal claw-free graph $G = (V, E)$. Then for all $v \in V$ the induced subtree $T[\mathcal{M}_v]$ is a path in T.*

Proof. Let $G = (V, E) \in$ CCF and let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of G. Let us assume there exists a vertex $v \in V$ such that the graph $T[\mathcal{M}_v]$ is not a path in T. As $T[\mathcal{M}_v]$ is a subtree of T, there exists a max clique $B \in \mathcal{M}_v$ such that B has degree at least 3. Let $A_1, A_2, A_3 \in \mathcal{M}_v$ be three distinct neighbors of B in $T[\mathcal{M}_v]$. Since A_i and B are distinct max cliques, there exists a vertex $a_i \in A_i \setminus B$, and for each $i \in [3]$, we have $A_i \in \mathcal{M}_{a_i}$, $B \notin \mathcal{M}_{a_i}$ and $T[\mathcal{M}_{a_i}]$ is connected. As T is a tree, A_1, A_2, and A_3 are all in different connected components of $T[\mathcal{M} \setminus \{B\}]$. Therefore, $\mathcal{M}_{a_i} \cap \mathcal{M}_{a_{i'}} = \emptyset$ for all $i, i' \in [3]$ with $i \neq i'$. Now, $\{v, a_1, a_2, a_3\}$ induces a claw in G, which contradicts G being claw-free: For all $i \in [3]$, there is an edge between v and a_i, because $v, a_i \in A_i$. To show that vertices a_i and $a_{i'}$ are not adjacent for $i \neq i'$, let us assume the opposite. If a_i and $a_{i'}$ are adjacent, then there exists a max clique M containing a_i and $a_{i'}$. Thus, $\mathcal{M}_{a_i} \cap \mathcal{M}_{a_{i'}} \neq \emptyset$, a contradiction. ◀

The following lemmas help us to show in Corollary 3.5 that the clique tree of a connected chordal claw-free graph is unique. This is a property that does not hold for unconnected chordal (claw-free) graphs in general.

▶ **Lemma 3.2.** *Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal claw-free graph $G = (V, E)$. Further, let $v \in V$, and let A_1, A_2, A_3 be distinct max cliques in $\mathcal{M}_v$. Then A_2 lies between A_1 and A_3 on the path $T[\mathcal{M}_v]$ if and only if $A_2 \subseteq A_1 \cup A_3$.*

Proof. Let $G = (V, E) \in$ CCF and $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of G. Further, let $v \in V$, and let $A_1, A_2, A_3 \in \mathcal{M}_v$ be distinct max cliques. First, let $A_2 \subseteq A_1 \cup A_3$, and let us assume that, w.l.o.g., A_1 lies between A_2 and A_3. Then $A_2 \cap A_3 \subseteq A_1$ according to the clique intersection property. Further, $A_2 \subseteq A_1 \cup A_3$ implies that $A_2 \setminus A_3 \subseteq A_1$. It follows that $A_2 \subseteq A_1$, which is a contradiction to A_1 and A_2 being distinct max cliques.

Now let max clique A_2 lie between A_1 and A_3 on the path $T[\mathcal{M}_v]$, and let us assume that there exists a vertex $a_2 \in A_2 \setminus (A_1 \cup A_3)$. Let $P = B_1, \ldots, B_l$ be the path $T[\mathcal{M}_v]$ (Lemma 3.1).

W.l.o.g., let $A_i = B_{j_i}$ for all $i \in [3]$ where $j_1, j_2, j_3 \in [l]$ with $j_1 < j_2 < j_3$. Further, let $A'_1 := B_{j_1+1}$ and $A'_3 := B_{j_3-1}$, and let $a_1 \in A_1 \setminus A'_1$ and $a_3 \in A_3 \setminus A'_3$. Similar to the proof of Lemma 3.1, we obtain that $\{v, a_1, a_2, a_3\}$ induces a claw in G, a contradiction. ◀

▶ **Lemma 3.3.** *Let $T_1 = (\mathcal{M}, \mathcal{E}_1)$ and $T_2 = (\mathcal{M}, \mathcal{E}_2)$ be clique trees of a chordal claw-free graph $G = (V, E)$. Then for every $v \in V$ we have $T_1[\mathcal{M}_v] = T_2[\mathcal{M}_v]$.*

Proof. Let $G = (V, E) \in \mathsf{CCF}$ and let $T_1 = (\mathcal{M}, \mathcal{E}_1)$ and $T_2 = (\mathcal{M}, \mathcal{E}_2)$ be clique trees of G. Let $v \in V$. According to Lemma 3.1, $T_1[\mathcal{M}_v]$ and $T_2[\mathcal{M}_v]$ are paths in T_1 and T_2, respectively. Let us assume there exist distinct max cliques $A, B \in \mathcal{M}_v$ such that, without loss of generality, A, B are adjacent in $T_1[\mathcal{M}_v]$ but not adjacent in $T_2[\mathcal{M}_v]$. As A and B are not adjacent in $T_2[\mathcal{M}_v]$, there exists a max clique $C \in \mathcal{M}_v$ that lies between A and B on the path $T_2[\mathcal{M}_v]$. Thus, $A \cap B \subseteq C$ according to the clique-intersection property. Since max cliques A and B are adjacent in $T_1[\mathcal{M}_v]$, either A lies between B and C, or B lies between A and C on the path $T_1[\mathcal{M}_v]$. W.l.o.g., let A lie between B and C on the path $T_1[\mathcal{M}_v]$. Then $A \subseteq B \cup C$ by Lemma 3.2. Thus, we have $A \setminus B \subseteq C$. Since $A \cap B \subseteq C$, this yields that $A \subseteq C$, which is a contradiction to A and C being distinct max cliques. ◀

▶ **Lemma 3.4.** *Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a connected chordal claw-free graph $G = (V, E)$. Then*

$$T = \bigcup_{v \in V} T[\mathcal{M}_v].$$

Proof. Let $G = (V, E)$ be a connected chordal claw-free graph and $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of G. Clearly, the graphs T and $T' := \bigcup_{v \in V} T[\mathcal{M}_v]$ have the same vertex set, and T' is a subgraph of the tree T. In order to prove that $T = T'$, we show that T' is connected.

For all vertices $v \in V$, the graph $T'[\mathcal{M}_v]$ is connected because $T[\mathcal{M}_v]$ is connected. For each edge $\{u, v\} \in E$ of the graph G, there exists a max clique that contains u and v, and therefore, we have $\mathcal{M}_u \cap \mathcal{M}_v \neq \emptyset$. Hence, $T'[\mathcal{M}_u \cup \mathcal{M}_v]$ is connected for every edge $\{u, v\} \in E$. Since G is connected, it follows that $T'[\bigcup_{v \in V} \mathcal{M}_v]$ is connected. Clearly, $\bigcup_{v \in V} \mathcal{M}_v = \mathcal{M}$. Consequently, the graph T' is connected. ◀

As a direct consequence of Lemma 3.3 and Lemma 3.4 we obtain the following corollary. It follows that each connected chordal claw-free graph has a unique clique tree.

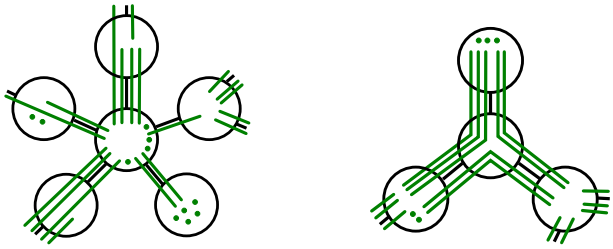
▶ **Corollary 3.5.** *Let T_1 and T_2 be clique trees of a connected chordal claw-free graph G. Then $T_1 = T_2$.*

In the following let $G = (V, E)$ be a connected chordal claw-free graph and let $T_G = (\mathcal{M}, \mathcal{E})$ be its clique tree.

Let B be a max clique of G. If for all $v \in B$ max clique B is an end of path $T_G[\mathcal{M}_v]$, we call B a *star clique*. Thus, B is a star clique if, and only if, every vertex in B is contained in at most one neighbor of B in T_G. A picture of a star clique can be found in Figure 2a. Clearly, every max clique of degree 1, i.e. every leaf, of clique tree T_G is a star clique.

A max clique B of degree 3 is called a *fork clique* if for every $v \in B$ there exist two neighbors A, A' of B with $A \neq A'$ such that $\mathcal{M}_v = \{B, A, A'\}$, and for all neighbors A, A' of B with $A \neq A'$ there exists a vertex $v \in B$ with $\mathcal{M}_v = \{B, A, A'\}$. Figure 2b shows a sketch of a fork clique. Note that two fork cliques cannot be adjacent.

The following lemma provides more information about the structure of the clique tree of a connected chordal claw-free graph. A proof can be found in Appendix A.



(a) A star clique **(b)** A fork clique

Figure 2 A star clique and a fork clique.

▶ **Lemma 3.6.** *Let $B \in \mathcal{M}$. If the degree of B in clique tree T_G is at least 3, then B is a star clique or a fork clique.*

▶ **Corollary 3.7.** *Let $B \in \mathcal{M}$ be a fork clique. Then every neighbor of B in clique tree T_G is a star clique.*

Proof. Let us assume max clique A is a neighbor of fork clique B, and A is not a star clique. Then the degree of A is at least 2. As A cannot be a fork clique, Lemma 3.6 implies that A has degree 2. Since B is a fork clique, there does not exist a vertex $v \in A$ that is contained in B and the other neighbor of A. Thus, A is a star clique, a contradiction. ◀

4 Defining the Supplemented Clique Tree in STC+C

In this section we define the supplemented clique tree of a connected chordal claw-free graph G. We obtain the supplemented clique tree by transferring the clique tree T_G into a directed tree and including some of the structural information about each max clique into the directed clique tree by means of an LO-coloring. We show that there exists a parameterized STC+C-transduction that defines for each connected chordal claw-free graph and every tuple of suitable parameters an isomorphic copy of such a supplemented clique tree. In order to do this, we first present (parameterized) transductions for the clique tree and the directed clique tree. Throughout this section we let $\bar{x}, \bar{y}$ and $\bar{y}'$ be triples of structure variables.

In a first step we present a transduction $\Theta = (\theta_U(\bar{y}), \theta_{\approx}(\bar{y}, \bar{y}'), \theta_E(\bar{y}, \bar{y}'))$ that defines for each connected chordal claw-free graph G a tree isomorphic to the clique tree of G.

In the following, let $G = (V, E)$ be a chordal claw-free graph, and let $\mathcal{M}$ be the set of max cliques of G. A triple $\bar{b} = (b_1, b_2, b_3) \in V^3$ *spans* a max clique $A \in \mathcal{M}$ if A is the only max clique that contains the vertices b_1, b_2 and b_3. Thus, $\bar{b}$ spans max clique $A \in \mathcal{M}$ if and only if $\mathcal{M}_{b_1} \cap \mathcal{M}_{b_2} \cap \mathcal{M}_{b_3} = \{A\}$. We call $\bar{b} \in V^3$ a *spanning triple* of G if $\bar{b}$ spans a max clique. We use spanning triples to represent max cliques. Note that this concept was already used in [20] and [14] to represent max cliques of interval graphs.

▶ **Lemma 4.1.** *Every max clique of a chordal claw-free graph is spanned by a triple of vertices.*

Proof. Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal claw-free graph G. Let $B \in \mathcal{M}$ and let $v \in B$. By Lemma 3.1, the induced subgraph $T[\mathcal{M}_v]$ is a path $P = B_1, \ldots, B_l$. Let $B = B_i$. If $i > 1$, let u be a vertex in $B \setminus B_{i-1}$, and let w be a vertex in $B \setminus B_{i+1}$ if $i < l$. We let $u = v$ if $i = 1$, and we let $w = v$ if $i = l$. Then (u, v, w) spans max clique B: Clearly, $u, v, w \in B$. It remains to show, that there does not exist a max clique $A \in \mathcal{M}$ with $A \neq B$ and $u, v, w \in A$. Let us suppose such a max clique A exists. Since $v \in A$, max clique A is

a vertex on path P. W.l.o.g., let $A = B_j$ for $j < i$. According to the clique intersection property, we have $u \in A \cap B \subseteq B_{i-1}$, a contradiction. ◀

As a direct consequence of Lemma 4.1, there exists an at most cubic number of max cliques in a chordal claw-free graph.

The following observations contain properties that help us to define the transduction Θ. Proofs can be found in [16].

▶ **Observation 4.2.** *Let $G = (V, E)$ be a chordal claw-free graph. Let $\bar{v} = (v_1, v_2, v_3) \in V^3$. Then $\bar{v}$ is a spanning triple of G if, and only if, $\tilde{v}$ is a clique and $\{w_1, w_2\} \in E$ for all vertices $w_1, w_2 \in N(v_1) \cap N(v_2) \cap N(v_3)$ with $w_1 \neq w_2$.*

From the characterization of spanning triples in Observation 4.2, it follows that there exists an FO-formula $\theta_U(\bar{y})$ that is satisfied by a chordal claw-free graph $G = (V, E)$ and a triple $\bar{v} \in V^3$ if and only if $\bar{v}$ is a spanning triple of G.

▶ **Observation 4.3.** *Let $G = (V, E)$ be a chordal claw-free graph. Let A be a max clique of G, and let the triple $\bar{v} = (v_1, v_2, v_3) \in V^3$ span A. Then $w \in A$ if, and only if, $w \in \tilde{v}$ or $\{w, v_j\} \in E$ for all $j \in [3]$.*

Observation 4.3 yields that there further exists an FO-formula $\varphi_{\mathcal{M}}(\bar{y}, z)$ that is satisfied for $\bar{v} \in V^3$ and $w \in V$ in a chordal claw-free graph $G = (V, E)$ if, and only if, $\bar{v}$ spans a max clique A and $w \in A$. We can use this formula to obtain an FO-formula $\theta_{\approx}(\bar{y}, \bar{y}')$ such that for all chordal claw-free graphs $G = (V, E)$ and all triples $\bar{v}, \bar{v}' \in V^3$ we have $G \models \theta_{\approx}(\bar{v}, \bar{v}')$ if, and only if, $\bar{v}$ and $\bar{v}'$ span the same max clique.

In the following we consider connected chordal claw-free graphs G. The next observation is a direct consequence of Lemma 3.4 and Lemma 3.2.

▶ **Observation 4.4.** *Let $G = (V, E)$ be a connected chordal claw-free graph, and $T_G = (\mathcal{M}, \mathcal{E})$ be the clique tree of G. Let $A, B \in \mathcal{M}$. Max cliques A and B are adjacent in T_G if, and only if, there exists a vertex $v \in V$ such that $v \in A \cap B$ and for all $C \in \mathcal{M}$ with $v \in C$ we have $C \not\subseteq A \cup B$.*

It follows from Observation 4.4 that there exists an FO-formula $\theta_E(\bar{y}, \bar{y}')$ that is satisfied for triples $\bar{v}, \bar{v}' \in V^3$ in a connected chordal claw-free graph $G = (V, E)$ if, and only if, $\bar{v}$ and $\bar{v}'$ span adjacent max cliques.

It is not hard to see that $\Theta = (\theta_U, \theta_{\approx}, \theta_E)$ is an FO-transduction that defines for each connected chordal claw-free graph G a tree isomorphic to the clique tree of G.

▶ **Lemma 4.5.** *There exists an FO-transduction Θ such that $\Theta[G] \cong T_G$ for all $G \in$ con-CCF.*

Now we transfer the clique tree into a directed tree. Let R be a leaf of the clique tree T_G. We transform T_G into a directed tree by rooting T_G at max clique R. We denote the resulting directed clique tree by $T_G^R = (\mathcal{M}, \mathcal{E}_R)$. Since R is a leaf of T_G, the following corollary is an immediate consequence of Lemma 3.6.

▶ **Corollary 4.6.** *Let A be a max clique of a connected chordal claw-free graph G. Then A is a vertex with at least two children in T_G^R only if A is a star clique or a fork clique.*

In the following we show that there exists a parameterized STC-transduction $\Theta'(\bar{x})$ which defines an isomorphic copy of T_G^R for each connected chordal claw-free graph G and triple $\bar{r} \in V^3$ that spans a leaf R of T_G.

Clearly, we can define an FO-formula $\theta'_{\mathrm{dom}}(\bar{x})$ such that for all connected chordal claw-free graphs G and $\bar{r} \in V^3$ we have $G \models \theta'_{\mathrm{dom}}(\bar{r})$ if, and only if, $\bar{r} \in V^3$ spans a leaf of

T_G. Then θ'_{dom} defines the triples of parameters of transduction $\Theta'(\bar{x})$. Further, we let $\theta'_U(\bar{x},\bar{y}) := \theta_U(\bar{y})$ and $\theta'_{\approx}(\bar{x},\bar{y},\bar{y}') := \theta_{\approx}(\bar{y},\bar{y}')$. Finally, we let $\theta'_E(\bar{x},\bar{y},\bar{y}')$ be satisfied for triples $\bar{r},\bar{v},\bar{v}' \in V^3$ in a connected chordal claw-free graph $G=(V,E)$ if, and only if, $\bar{r}$, $\bar{v}$ and $\bar{v}'$ span max cliques R, A and A', respectively, and (A,A') is an edge in T_G^R. Note that (A,A') is an edge in T_G^R precisely if $\{A,A'\}$ is an edge in T_G and there exists a path between R and A in T_G after removing A'. Thus, formula θ'_E can be constructed in STC. We let $\Theta'(\bar{x}) := (\theta'_{\text{dom}}(\bar{x}),\theta'_U(\bar{x},\bar{y}),\theta'_{\approx}(\bar{x},\bar{y},\bar{y}'),\theta'_E(\bar{x},\bar{y},\bar{y}'))$, and conclude:

▶ **Lemma 4.7.** *There exists a parameterized* STC*-transduction* $\Theta'(\bar{x})$ *such that* $\text{Dom}(\Theta'(\bar{x}))$ *is the set of all pairs* $(G,\bar{r})$ *where* $G=(V,E) \in$ con-CCF *and* $\bar{r} \in V^3$ *spans a leaf* R *of* T_G*, and* $\Theta'[G,\bar{r}] \cong T_G^R$ *for all* $(G,\bar{r}) \in \text{Dom}(\Theta'(\bar{x}))$ *where* $\bar{r}$ *spans the max clique* R *of* G*.*

We now equip each max clique of the directed clique tree T_G^R with structural information. We do this by coloring the directed clique tree T_G^R with an LO-coloring. An LO-color is a binary relation on a linearly ordered set of basic color elements. Into each LO-color, we encode three numbers. Isomorphisms of LO-colored trees preserve the information that is encoded in the LO-colors. Thus, an LO-colored tree and its canon contain the same numbers encoded in their LO-colors. We call this LO-colored directed clique tree a supplemented clique tree. More precisely, let $G \in$ con-CCF and let R be a leaf of the clique tree T_G of G, then the *supplemented clique tree* S_G^R is the 5-tuple $(\mathcal{M},\mathcal{E}_R,[0,|V|],\leq_{[0,|V|]},L)$ where

- $(\mathcal{M},\mathcal{E}_R)$ is the directed clique tree T_G^R of G,
- $\leq_{[0,|V|]}$ is the natural linear order on the set of basic color elements $[0,|V|]$,
- $L \subseteq \mathcal{M} \times [0,|V|]^2$ is the ternary color relation where
  - $(A,0,n) \in L$ iff n is the number of vertices in A that are not in any child of A in T_G^R,
  - $(A,1,n) \in L$ iff n is the number of vertices that are contained in A and in the parent of A in T_G^R if $A \neq R$, and $n=0$ if $A=R$,
  - $(A,2,n) \in L$ iff n is the number of vertices in A that are in two children of A in T_G^R.[8]

In its structural representation the supplemented clique tree S_G^R corresponds to the 6-tuple $(\mathcal{M} \dot{\cup} [0,|V|],\mathcal{M},\mathcal{E}_R,[0,|V|],\leq_{[0,|V|]},L)$.

The properties encoded in the colors of the max cliques are easily expressible in STC+C. Therefore, we can extend the parameterized STC-transduction $\Theta'(\bar{x})$ to a parameterized STC+C-transduction $\Theta''(\bar{x})$ that defines an LO-colored graph isomorphic to S_G^R for every connected chordal claw-free graph G and triple $\bar{r} \in V^3$ that spans a leaf R of T_G. More details on how to construct $\Theta''(\bar{x})$ can be found in [16].

▶ **Lemma 4.8.** *There is a parameterized* STC+C*-transduction* $\Theta''(\bar{x})$ *such that* $\text{Dom}(\Theta''(\bar{x}))$ *is the set of all pairs* $(G,\bar{r})$ *where* $G=(V,E) \in$ con-CCF *and* $\bar{r} \in V^3$ *spans a leaf* R *of* T_G*, and* $\Theta''[G,\bar{r}] \cong S_G^R$ *for all* $(G,\bar{r}) \in \text{Dom}(\Theta''(\bar{x}))$ *where* $\bar{r}$ *spans the max clique* R *of* G*.*

5 Canonization

In this section we prove that there exists a parameterized $\text{LREC}_=$-canonization of the class con-CCF of connected chordal claw-free graphs. Then Proposition 2.2 implies that there also exists one for the class CCF of chordal claw-free graphs, and we obtain our main result:

▶ **Theorem 5.1.** *The class of chordal claw-free graphs admits* $\text{LREC}_=$*-definable canonization.*

[8] Let A be a max clique and n be the number of vertices in A that are in two children of A in T_G^R. Corollary 4.6 implies that A is a fork clique if and only if $n>0$.

As a consequence of Proposition 2.6 and the logarithmic-space data complexity of $\mathsf{LREC}_=$ we obtain the following corollaries.

▶ **Corollary 5.2.** $\mathsf{LREC}_=$ *captures* LOGSPACE *on the class of chordal claw-free graphs.*

▶ **Corollary 5.3.** *There exists a logarithmic-space canonization algorithm for the class of chordal claw-free graphs.*

Since $\mathsf{LREC}_=$ is contained in FP+C [14], Theorem 5.1 also implies that there exists an FP+C-canonization of the class of chordal claw-free graphs. We directly obtain (see e.g. [4]):

▶ **Corollary 5.4.** FP+C *captures* PTIME *on the class of chordal claw-free graphs.*

Now let us prove that there exists a parameterized $\mathsf{LREC}_=$-canonization of con-CCF. By Lemma 4.8 there exists a parameterized STC+C-, and therefore, $\mathsf{LREC}_=$-transduction $\Theta''(\bar{x})$ such that $\Theta''[G,\bar{r}]$ is isomorphic to the LO-colored tree S_G^R for all connected chordal claw-free graphs $G=(V,E)$ and all triples $\bar{r}\in V^3$ that span a leaf R of T_G. Further, there exists an $\mathsf{LREC}_=$-canonization Θ^{LO} of the class of LO-colored directed trees according to Theorem 2.3. In the following, we show that there also exists an $\mathsf{LREC}_=$-transduction Θ^K which defines for each canon $K(S_G^R)$ of a supplemented clique tree S_G^R of $G\in$ con-CCF the canon $K(G)$ of G. Then we can compose the (parameterized) $\mathsf{LREC}_=$-transductions $\Theta''(\bar{x})$, Θ^{LO} and Θ^K to obtain a parameterized $\mathsf{LREC}_=$-canonization of the class of connected chordal claw-free graphs (see [16]).

We let $\mathsf{LREC}_=[\{V,E,M,\trianglelefteq,L,\leq\},\{E,\leq\}]$-transduction $\Theta^K=(\theta_V(p),\theta_E(p,p'),\theta_\leq(p,p'))$ define for each canon $K(S_G^R)=(U_K,V_K,E_K,M_K,\trianglelefteq_K,L_K,\leq_K)$ of a supplemented clique tree of $G\in$ con-CCF an ordered isomorphic copy $K(G)=(V_K',E_K',\leq_K')$ of $G=(V,E)$. We let V_K' be the set $[|V|]$, and $\leq_K'$ be the natural linear order on $[|V|]$. As $[0,|V|]$ is the set of basic color elements of S_G^R, the set M_K of basic color elements of the canon $K(S_G^R)$ contains exactly $|V|+1$ elements. Hence, we can easily define the vertex set of $K(G)$ by counting the number of basic color elements of $K(S_G^R)$. We let $\varphi_V(p):=\exists q\left(p\leq q\wedge p\neq 0\wedge \#x\,M(x)=q\right)$. Further, we let $\theta_\leq(p,p'):=p\leq p'$. In order to show that there exists an $\mathsf{LREC}_=$-formula $\theta_E(p,p')$, which defines the edge relation of $K(G)$, we exploit the property that $\mathsf{LREC}_=$ captures LOGSPACE on ordered structures (Corollary 2.5), and show that there exists a logarithmic-space algorithm that computes the edge relation of $K(G)$, instead. In order to do this, we present a logarithmic-space algorithm that outputs the max cliques of $K(G)$. As every edge is a subset of some max clique and every two distinct vertices in a max clique are adjacent, such a logarithmic-space algorithm can easily be extended to a logarithmic-space algorithm that decides whether a pair of numbers is an edge of $K(G)$.

▶ **Lemma 5.5.** *There exists a logarithmic-space algorithm that, given the canon $K(S_G^R)$ of a supplemented clique tree of $G\in$* con-CCF*, computes the set of max cliques of the canon $K(G)$ of G.*

In the following, we sketch the idea of the algorithm. A detailed proof of Lemma 5.5 can be found in [16].

The algorithm performs a post-order tree traversal[9] on the underlying tree of the canon $K(S_G^R)$ of the supplemented clique tree S_G^R. Let $m_1,\ldots,m_{|\mathcal{M}|}$ be the respective post-order

[9] In a *tree traversal*, we visit every vertex of the tree exactly once. We obtain the *post-order traversal* (see e.g. [23]) of an ordered directed tree T with root r recursively as follows: Let d be the out-degree of r. For all $i\in\{1,\ldots,d\}$, in increasing order, perform a post-order traversal on the subtree rooted at child i of the root. Afterward, visit r. For example in [22], it is shown that there exists a logarithmic-space algorithm for depth-first tree traversal. By selecting all vertices that are not followed by the move **down** in the depth-first traversal in [22], we obtain the post-order traversal sequence in logarithmic space.

traversal sequence of the vertices. Each vertex $m_k \in V_K$ of the canon $K(S_G^R)$ corresponds to a vertex, i.e. a max clique $A_k \in \mathcal{M}$, in the supplemented clique tree S_G^R. We call $A_1, \ldots, A_{m_{|\mathcal{M}|}}$ the *transferred traversal sequence*. For all $k \in \{1, \ldots, |\mathcal{M}|\}$, starting with $k = 1$, the algorithm constructs for $m_k \in V_K$ a copy $B_{m_k} \subseteq [|V|]$ of A_k.

From the information encoded in the colors, we know the number of vertices in A_k that are not in any max clique that occurs before A_k in the transferred traversal sequence. For these vertices, we add the smallest numbers of $[|V|]$ to B_{m_k} that were not already used. Further, we need to know how many vertices of A_k are in a max clique A_i that occurs before A_k in the transferred traversal sequence, and what numbers these vertices were assigned to. These numbers have to be added to B_{m_k} as well.

Now, if A_k is a fork clique (the information of whether vertex m_k corresponds to a fork clique A_k is encoded in the color of m_k), then we know the vertices of A_k are all contained in at least one of its two children, which occur before A_k in the transferred traversal sequence; and we can use the information in the colors of m_k and its children to find out the number of vertices in A_k that are contained in the first child, the second child and both children of A_k, respectively.

If A_k is not a fork clique, then the vertices in A_k that are in max cliques that occur before A_k within the transferred traversal sequence are precisely the vertices in the pairwise intersection of A_k with its children, and the intersection of A_k with its sibling if A_k is the second child of a fork clique. Note that these intersections are disjoint sets of vertices because A_k is a star clique, or otherwise only has one child (Corollary 4.6) and cannot be the second child of a fork clique by Corollary 3.7. Again we can use the information within the colors to find out the number of vertices in A_k that are contained in the respective child of A_k, and the number of vertices in the intersection of A_k and its sibling if A_k is the second child of a fork clique.

Therefore, in both cases, all vertices of A_k that are in a max clique A_i that occurs before A_k in the transferred traversal sequence, are contained in the children of A_k, or the first child of a fork clique if A_k is the second one.

For the numbers in each max clique B_{m_j} (where m_j does not corresponds to the second child of a fork clique), we maintain the property that if a number $l \in B_{m_j}$ is contained in more ancestors of B_{m_j} than a number $l' \in B_{m_j}$, then $l > l'$. Thus, if B_{m_j} is a child of a max clique $B_{m_{j'}}$, then the intersection $B_{m_j} \cap B_{m_{j'}}$ contains precisely the $|B_{m_j} \cap B_{m_{j'}}|$ largest numbers of B_{m_j}. We can use this property to determine the numbers in B_{m_i} that have to be included into B_{m_k} if m_i is a child of m_k and also if m_i corresponds to the first child of a fork clique and m_k to the second one. Note that we do not have to remember the max cliques $B_{m_1}, \ldots, B_{m_{k-1}}$ as we can recompute all values that are necessary to obtain the numbers that have to be included into B_{m_k}. The recomputation can be done in logarithmic space. As a consequence, we obtain an algorithm that computes the max cliques of the canon $K(G)$ in logarithmic space.

6 Conclusion

Currently, there exist hardly any logical characterizations of LOGSPACE on non-trivial natural classes of unordered structures. The only ones previously presented are that $\mathsf{LREC}_=$ captures LOGSPACE on (directed) trees and interval graphs [13, 14]. By showing that $\mathsf{LREC}_=$ captures LOGSPACE also on the class of chordal claw-free graphs, we contribute a further characterization of LOGSPACE on an unordered graph class. It would be interesting to investigate further classes of unordered structures such as the class of planar graphs or classes of graphs of bounded treewidth. The author conjectures that $\mathsf{LREC}_=$ captures LOGSPACE on the class of all planar graphs that are equipped with an embedding.

Our main result, which states that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization, does not only imply that $\mathsf{LREC}_=$ captures LOGSPACE on this graph class, but also that there exists a logarithmic-space canonization algorithm for the class of chordal claw-free graphs. Hence, the isomorphism and automorphism problem for this graph class is solvable in logarithmic space.

Further, we make a contribution to the investigation of PTIME's characteristics on restricted classes of graphs. It follows from our main result that there is an FP+C-canonization of the class of chordal claw-free graphs. As a consequence, FP+C captures PTIME on this graph class. Thus, we add the class of chordal claw-free graphs to the (so far) short list of graph classes that are not closed under taking minors and on which PTIME is captured.

Acknowledgements. The author wants to thank Nicole Schweikardt and the reviewers for helpful comments that contributed to improving the paper.

References

1. J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In A. George, J. R. Gilbert, and J. W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*, pages 1–29. Springer New York, 1993.
2. P. Buneman. A characterisation of rigid circuit graphs. *Discrete Math.*, 9:205–212, 1974.
3. J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.
4. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1999.
5. K. Etessami and N. Immerman. Tree canonization and transitive closure. *Information and Computation*, 157(1–2):2–24, 2000.
6. R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R.M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings 7*, pages 43–73, 1974.
7. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.
8. M. Grohe. Fixed-point logics on planar graphs. In *LICS*, pages 6–15, 1998.
9. M. Grohe. Definable tree decompositions. In *LICS*, pages 406–417, 2008.
10. M. Grohe. Fixed-point definability and polynomial time on chordal graphs and line graphs. In A. Blass, N. Dershowitz, and W. Reisig, editors, *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, pages 328–353, 2010.
11. M. Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. In *LICS*, 2010.
12. M. Grohe. Descriptive complexity, canonisation, and definable graph structure theory, 2013. [Online; accessed 2017-03-31]. URL: `http://lii.rwth-aachen.de/images/Mitarbeiter/pub/grohe/cap/all.pdf`.
13. M. Grohe, B. Grußien, A. Hernich, and B. Laubner. L-recursion and a new logic for logarithmic space. In *CSL*, pages 277–291, 2011.
14. M. Grohe, B. Grußien, A. Hernich, and B. Laubner. L-recursion and a new logic for logarithmic space. *Logical Methods in Computer Science*, 9(1), 2012.
15. M. Grohe and J. Mariño. Definability and descriptive complexity on databases of bounded tree-width. In *ICDT*, pages 70–82, 1999.
16. B. Grußien. *Capturing Polynomial Time and Logarithmic Space using Modular Decompositions and Limited Recursion*. PhD thesis, Humboldt-Universität zu Berlin, 2016.
17. B. Grußien. Capturing polynomial time using modular decomposition. In *LICS*, 2017.
18. N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

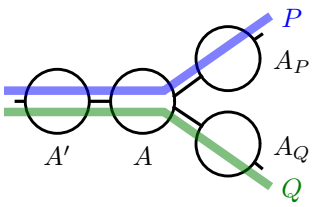


Figure 3 A fork of P and Q.

19 N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16:760–778, 1987.
20 B. Laubner. Capturing polynomial time on interval graphs. In *LICS*, pages 199–208, 2010.
21 B. Laubner. *The Structure of Graphs and New Logics for the Characterization of Polynomial Time*. PhD thesis, Humboldt-Universität zu Berlin, 2011.
22 S. Lindell. A logspace algorithm for tree canonization. In *STOC*, pages 400–404, 1992.
23 R. Sedgewick. *Algorithms in Java: Parts 1-4*. Addison-Wesley, 3rd edition, 2002.
24 M. Y. Vardi. The complexity of relational query languages. In *STOC*, pages 137–146, 1982.
25 J. R. Walter. *Representations of Rigid Cycle Graphs*. PhD thesis, Wayne State University, 1972.

A Proof of Lemma 3.6

In order to prove Lemma 3.6, we further analyze the structure of the clique tree. Throughout this section, we let $G = (V, E)$ be a connected chordal claw-free graph and $T_G = (\mathcal{M}, \mathcal{E})$ be its clique tree.

The following corollary follows directly from Lemma 3.2.

▶ **Corollary A.1.** *Let $v \in V$. Then for all $w \in V \setminus \{v\}$ the graph $T_G[\mathcal{M}_v \setminus \mathcal{M}_w]$ is connected.*[10]

Proof. Let $v \in V$ and let $w \in V \setminus \{v\}$. Let $P = A_1, \ldots, A_l$ be the path $T_G[\mathcal{M}_v]$, and let us assume $T_G[\mathcal{M}_v \setminus \mathcal{M}_w]$ is not connected. Then there exist $i, j, k \in [l]$ with $i < j < k$ such that $A_i, A_k \in \mathcal{M}_v \setminus \mathcal{M}_w$ and $A_j \in \mathcal{M}_w$. By Lemma 3.2 we have $A_j \subseteq A_i \cup A_k$. Thus, vertex $w \in A_j$ is also contained in A_i or A_k, a contradiction. ◀

Let P and Q be two paths in T_G. We call $(A', A, \{A_P, A_Q\}) \in V^2 \times \binom{V}{2}$ a *fork* of P and Q, if $P[\{A', A, A_P\}]$ and $Q[\{A', A, A_Q\}]$ are induced subpaths of length 3 of P and Q, respectively, and neither A_P occurs in Q nor A_Q occurs in P. Figure 3 shows a fork of paths P and Q. We say P and Q *fork (in B)* if there exists a fork $(A', A, \{A_P, A_Q\})$ of P and Q (with $A = B$).

▶ **Lemma A.2.** *Let $v, w \in V$. If the paths $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ fork, then $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ are paths of length 3.*

Proof. Let $v, w \in V$. Clearly, if $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ fork, then they must be paths of length at least 3. It remains to prove that their length is at most 3. For a contradiction, let us assume the length of $T_G[\mathcal{M}_v]$ is at least 4. Let $(A_1, B, \{A_2, A_2'\})$ be a fork of $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ where $A_2 \in \mathcal{M}_v \setminus \mathcal{M}_w$ and $A_2' \in \mathcal{M}_w \setminus \mathcal{M}_v$.

First let us assume there exists a max clique $A_0 \in \mathcal{M}_v$ such that $P = A_0, A_1, B, A_2$ is a subpath of $T_G[\mathcal{M}_v]$ of length 4. According to Corollary A.1, the graph $T_G[\mathcal{M}_v \setminus \mathcal{M}_w]$

[10] We define the empty graph as connected.

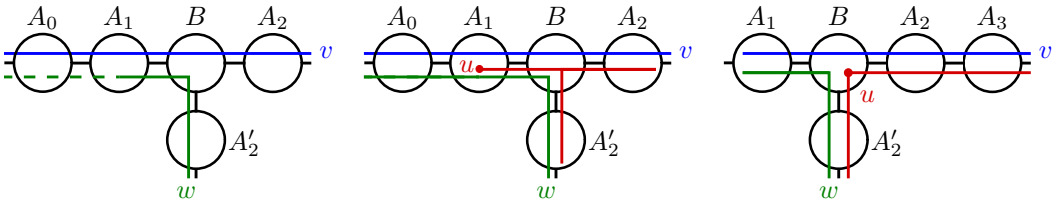


Figure 4 Illustrations for the proof of Lemma A.2.

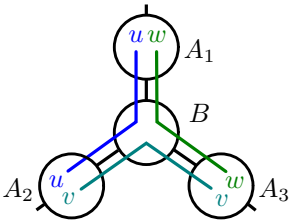


Figure 5 A fork triangle.

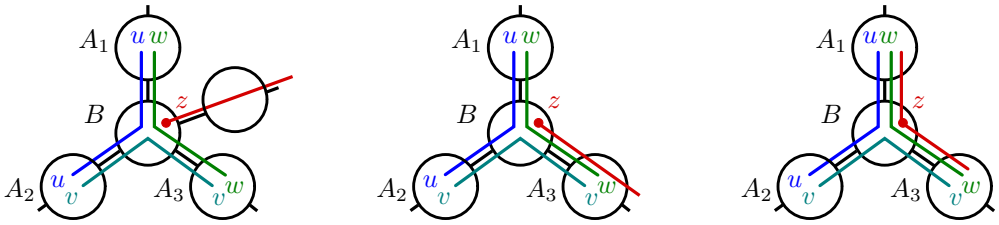
is connected. Thus, we have $A_0 \in \mathcal{M}_w$ (see Figure 4a). Now A_0 and A_1 are distinct max cliques. Therefore, there exists a vertex $u \in A_1 \setminus A_0$. As P is a subpath of $T_G[\mathcal{M}_v]$ and $P' = A_0, A_1, B, A'_2$ is a subpath of $T_G[\mathcal{M}_w]$, vertex u is not only contained in A_1 but also in B, A_2 and A'_2 by Lemma 3.2 (see Figure 4b). As a consequence, $T_G[\mathcal{M}_u]$ is not a path, a contradiction to Lemma 3.1.

Next, let us assume there exists a max clique $A_3 \in \mathcal{M}_v$ such that $P = A_1, B, A_2, A_3$ is a subpath of $T_G[\mathcal{M}_v]$ of length 4. Further, $P' = A_1, B, A'_2$ is a subpath of $T_G[\mathcal{M}_w]$. As A_1 and B are max cliques, there exists a vertex $u \in B \setminus A_1$. By Lemma 3.2, vertex u is also contained in A_2, A_3 and A'_2 as shown in Figure 4c. Now let us consider the paths $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_u]$. $Q = A_3, A_2, B, A_1$ is a subpath of $T_G[\mathcal{M}_v]$, and $Q' = A_3, A_2, B, A'_2$ is a subpath of $T_G[\mathcal{M}_u]$. Clearly, $(A_2, B, \{A_1, A'_2\})$ is a fork of $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_u]$. According to the previous part of this proof, we obtain a contradiction. ◀

The max cliques $A_1, A_2, A_3 \in \mathcal{M}$ form a *fork triangle* around a max clique $B \in \mathcal{M}$ if A_1, A_2 and A_3 are distinct max cliques in the neighborhood of B and there exist vertices $u, v, w \in V$ such that $\mathcal{M}_u = \{A_1, B, A_2\}$, $\mathcal{M}_v = \{A_2, B, A_3\}$ and $\mathcal{M}_w = \{A_3, B, A_1\}$. We say that max clique $B \in \mathcal{M}$ has a *fork triangle* if there exist max cliques $A_1, A_2, A_3 \in \mathcal{M}$ that form a fork triangle around B. Figure 5 depicts a fork triangle around a max clique B. Clearly, if a max clique B has a fork triangle, then B is a vertex of degree at least 3 in T_G.

▶ **Lemma A.3.** *Let $v, w \in V$, and let $B \in \mathcal{M}$ be a max clique. If $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$ fork in B, then B has a fork triangle.*

Proof. Let $v, w \in V$, let $B \in \mathcal{M}$ be a max clique, and let $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$ fork in B. Then $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$ are paths of length 3 by Lemma A.2. Let $\mathcal{M}_u = \{A_2, B, A_1\}$ and $\mathcal{M}_v = \{A_2, B, A_3\}$ with $A_1 \neq A_3$. Since B and A_2 are max cliques, there exists a vertex $w \in B \setminus A_2$. Now, we can apply Lemma 3.2 to the paths $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$, and obtain that $w \in A_1$ and $w \in A_3$. As $T_G[\mathcal{M}_w]$ and $T_G[\mathcal{M}_u]$ fork, the path $T_G[\mathcal{M}_w]$ must be of length 3 by Lemma A.2. Thus, $\mathcal{M}_w = \{A_3, B, A_1\}$. Hence, A_1, A_2, A_3 form a fork triangle around B. ◀



(a) $|\mathcal{M}_z \cap \mathcal{W}| = 0$ **(b)** $|\mathcal{M}_z \cap \mathcal{W}| = 1$ **(c)** $|\mathcal{M}_z \cap \mathcal{W}| = 2$

Figure 6 Illustrations for the proof of Lemma A.4.

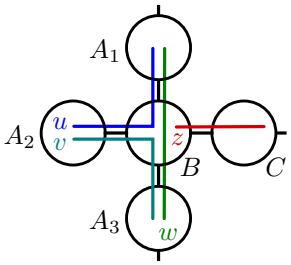


Figure 7 Illustration for the proof of Lemma A.5.

▶ **Lemma A.4.** *Let $z \in V$. If max clique $B \in \mathcal{M}_z$ has a fork triangle, then $|\mathcal{M}_z| = 3$ and B is in the middle of path $T_G[\mathcal{M}_z]$.*

Proof. Let $z \in V$, and let $B \in \mathcal{M}_z$ have a fork triangle. Then, there exist $u, v, w \in V$ and distinct neighbor max cliques A_1, A_2, A_3 of B such that $\mathcal{M}_u = \{A_1, B, A_2\}$, $\mathcal{M}_v = \{A_2, B, A_3\}$ and $\mathcal{M}_w = \{A_3, B, A_1\}$. Let $\mathcal{W}$ be the set $\{A_1, A_2, A_3\}$ of max cliques that form a fork triangle around B. Let us consider $|\mathcal{M}_z \cap \mathcal{W}|$. If $|\mathcal{M}_z \cap \mathcal{W}| \leq 1$, then $\mathcal{M}_z$ is a separating set of at least one of the paths $T_G[\mathcal{M}_u]$, $T_G[\mathcal{M}_v]$ or $T_G[\mathcal{M}_w]$ as shown in Figure 6a and 6b, and we have a contradiction to Corollary A.1. Clearly, we cannot have $|\mathcal{M}_z \cap \mathcal{W}| = 3$, since $T_G[\mathcal{M}_z]$ must be a path. It remains to consider $|\mathcal{M}_z \cap \mathcal{W}| = 2$, which is illustrated in Figure 6c. In this case, $T_G[\mathcal{M}_z]$ forks with one of the paths $T_G[\mathcal{M}_u]$, $T_G[\mathcal{M}_v]$ or $T_G[\mathcal{M}_w]$ in B, and must be of length 3 according to Lemma A.2. Obviously, B is in the middle of the path $T_G[\mathcal{M}_z]$. ◀

▶ **Lemma A.5.** *If max clique $B \in \mathcal{M}$ has a fork triangle, then the degree of B in T_G is 3.*

Proof. Let $B \in \mathcal{M}$ have a fork triangle. Thus, there exists vertices $u, v, w \in V$ and distinct neighbor max cliques A_1, A_2, A_3 of B such that $\mathcal{M}_u = \{A_1, B, A_2\}$, $\mathcal{M}_v = \{A_2, B, A_3\}$ and $\mathcal{M}_w = \{A_3, B, A_1\}$. Let us assume B is of degree at least 4. Let C be a neighbor of B in T_G that is distinct from A_1, A_2 and A_3. According to Lemma 3.4 there must be a vertex $z \in V$ such that $B, C \in \mathcal{M}_z$ (for an illustration see Figure 7). By Lemma A.4, we have $|\mathcal{M}_z| = 3$. W.l.o.g., let A_2 and A_3 be not contained in $\mathcal{M}_z$. Then $T_G[\mathcal{M}_v \setminus \mathcal{M}_z]$ is not connected, and we obtain a contradiction to Corollary A.1. ◀

▶ **Corollary A.6.** *If a max clique $B \in \mathcal{M}$ has a fork triangle, then B is a fork clique.*

Proof. Let B be a max clique that has a fork triangle. Then the degree of B is 3 by Lemma A.5. As B has a fork triangle, there exists a vertex $v \in B$ with $\mathcal{M}_v = \{B, A, A'\}$ for all neighbor max cliques A, A' of B with $A \neq A'$. Further, it follows from Lemma A.4

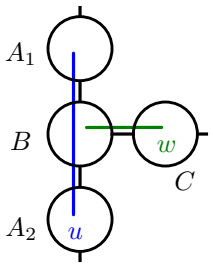


Figure 8 Illustration for the proof of Lemma 3.6.

that for every $v \in B$ there exist two neighbor max cliques A, A' of B with $A \neq A'$ such that $\mathcal{M}_v = \{B, A, A'\}$. ◀

▶ **Lemma 3.6** (restated). *Let $B \in \mathcal{M}$. If the degree of B in clique tree T_G is at least 3, then B is a star clique or a fork clique.*

Proof. Let B be a max clique of degree at least 3. Suppose B is not a star clique. Then there exists a vertex $u \in B$ and two neighbor max cliques A_1, A_2 of B in T_G that also contain vertex u. Let C be a neighbor of B with $C \neq A_1$ and $C \neq A_2$. Since $\{B, C\}$ is an edge of T_G, there must be a vertex $w \in V$ such that $B, C \in \mathcal{M}_w$ according to Lemma 3.4 (see Figure 8 for an illustration). By Corollary A.1, the graph $T_G[\mathcal{M}_u \setminus \mathcal{M}_w]$ must be connected. Thus, we have $A_1 \in \mathcal{M}_w$ or $A_2 \in \mathcal{M}_w$. Hence, $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_w]$ fork in B, and Lemma A.3 implies that B has a fork triangle. It follows from Corollary A.6 that B is a fork clique. ◀

The Model-Theoretic Expressiveness of Propositional Proof Systems

Erich Grädel[1], Benedikt Pago[2], and Wied Pakusa*[3]

1 **RWTH Aachen University, Aachen, Germany**
graedel@logic.rwth-aachen.de
2 **RWTH Aachen University, Aachen, Germany**
benedikt.pago@rwth-aachen.de
3 **University of Oxford, Oxford, UK**
wied.pakusa@cs.ox.ac.uk

Abstract

We establish new, and surprisingly tight, connections between propositional proof complexity and finite model theory. Specifically, we show that the power of several propositional proof systems, such as Horn resolution, bounded width resolution, and the polynomial calculus of bounded degree, can be characterised in a precise sense by variants of fixed-point logics that are of fundamental importance in descriptive complexity theory. Our main results are that *Horn resolution* has the same expressive power as *least fixed-point logic*, that *bounded width resolution* captures *existential least fixed-point logic*, and that the (monomial restriction of the) *polynomial calculus of bounded degree* solves precisely the problems definable in *fixed-point logic with counting*.

1998 ACM Subject Classification F.4.1. Mathematical Logic

Keywords and phrases Propositional proof systems, fixed-point logics, resolution, polynomial calculus, generalized quantifiers

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.27

1 Introduction

The question whether there exists an efficient proof system by means of which the validity of *arbitrary propositional formulas* can be verified via *proofs of polynomial size* is equivalent to the closure of NP under complementation. Since Cook and Reckhow [14] made the notion of an efficient propositional proof system precise, a huge body of research on the power of various propositional proof system has been established. In particular, we now have super-polynomial lower bounds on the proof complexity for quite strong proof systems, see [7, 25] for surveys on propositional proof complexity.

In this paper we study *polynomial-time variants* of propositional proof systems, which admit efficient proof search, resulting in proofs of polynomial size, such as restricted variants of resolution and the polynomial calculus. Recall that the resolution proof system Res takes as input a propositional formula φ in conjunctive normal form (CNF), and it refutes the satisfiability of φ if there is a derivation of the empty clause from φ. It is well-known that shortest resolution proofs can be of exponential size, so in general, we provably cannot search for resolution proofs in polynomial time. However, there are interesting restrictions of Res, such as Horn-Res (resolution restricted to Horn clauses) and bounded-width resolution

* The third author was supported by a DFG grant (PA 2962/1-1).

k-RES (resolution restricted to clauses of size $\leq k$) that do admit efficient proof search. Of course, unless P = NP, any proof system that admits efficient proof search is necessarily incomplete for full propositional logic. Nevertheless we can still prove interesting statements in such systems, and usually have completeness for relevant fragments of propositional logic, such as Horn-logic or 2-CNF. We can now try to solve algorithmic problems by reducing them to provability (or refutability) in some specific polynomial-time proof system, which, if it works successfully for all inputs, would give us a polynomial-time algorithm for the problem. Our goal is to understand how powerful this approach can be, depending on the specific proof system that we use.

Let us illustrate this by two concrete problems. First we consider *graph isomorphism*, a problem which is not known to be solvable in polynomial time although there is strong evidence that it is not NP-complete. Given two graphs $G = (V, E)$ and $H = (W, F)$ we ask whether there is a bijection $\pi\colon V \to W$ such that $\pi(E) = F$. Of course, this can easily be encoded as the satisfiability problem of a propositional CNF-formula. First, for each pair of vertices $v \in V$ and $w \in W$ we introduce a variable X_{vw} with the intended meaning that $X_{vw} = 1$ if $\pi(v) = w$. We add clauses $\bigvee_{w \in W} X_{vw}$ for every $v \in V$ and $\bigvee_{v \in V} X_{vw}$ for every $w \in W$ to ensure that every $v \in V$ has an image and every $w \in W$ has a preimage. Additionally we add for all $v_1, v_2 \in V$ and $w_1, w_2 \in W$ a clause $\neg(X_{v_1 w_1} \wedge X_{v_2 w_2})$ in case that $\{v_1 \mapsto w_1, v_2 \mapsto w_2\}$ is not a partial isomorphism. The resulting CNF-formula, denoted by $\mathrm{Iso}(G, H)$, is satisfiable if, and only if, the two graphs G and H are isomorphic. Following our reasoning from above, we can now use an efficient variant of resolution, or of a stronger proof system, and try to refute the satisfiability of the formula $\mathrm{Iso}(G, H)$. If this is possible, then G are H are not isomorphic. Unfortunately, if we do not find a proof, then we are stuck, because it might still be the case that G and H are not isomorphic, but our proof system is not strong enough to show this. Hence, we get an efficient, sound, but not necessarily complete graph isomorphism test. The question how successful this approach is when based on resolution was studied by Toran in [26]. Unfortunately, he proved that shortest resolution proofs for graph non-isomorphism can be of exponential size (even for graphs with colour class size four). More recently, Grohe and Berkholz showed that also in the stronger system polynomial calculus (PC) one cannot obtain small proofs for graph non-isomorphism [9, 10].

Our second example is directed graph reachability: Given a directed graph $G = (V, E)$ with two distinguished vertices $s, t \in V$, we want to know whether there is a path from s to t in G. Again, it is easy to encode this as a satisfiability problem in propositional logic, by taking the conjunction of all implication clauses $X_v \to X_w$, for all edges $(v, w) \in E$, together with the two clauses $1 \to X_s$ and $X_t \to 0$. Clearly the resulting formula $\mathrm{NonReach}(G, s, t)$ is unsatisfiable if, and only if, t is reachable from s in G. However, in clear contrast to the formulas $\mathrm{Iso}(G, H)$ from above, we can easily prove unsatisfiability for the formulas $\mathrm{NonReach}(G, s, t)$ in efficient variants of resolution such as HORN-RES and k-RES for $k \geq 2$.

Our two examples demonstrate the following: while certain problems, such as directed graph reachability, allow for small and efficient resolution proofs, other problems, such as the graph isomorphism problem, provably require proofs of super-polynomial size even in quite strong proof systems. This leads to the main question that we want to address in this paper: is there a *classification* for those problems which can be solved in fundamental polynomial-time propositional proof systems such as HORN-RES, k-RES and degree-k (monomial)-PC, denoted by MON-PC_k. It came as a surprise to us that there is, indeed, a very clear and tight classification of the power of all of these proof systems in terms of definability in important fixed-point logics which are well-studied in the area of descriptive complexity theory.

Before we can state our results in detail, we have to explain what we mean by saying that a problem, such as directed graph reachability, can be solved by a propositional proof system PROP. As usual, each decision problem can be identified with a membership problem "$\mathfrak{A} \in \mathcal{K}$?"

for some class of structures $\mathcal{K}$. For instance, the graph reachability problem from above is identified with the class $\mathcal{K}_{\text{Reach}} = \{(V, E, s, t) : \text{there is a path from } s \text{ to } t \text{ in } G = (V, E)\}$. Then we naturally want to say that a problem $\mathcal{K}$ can be solved by the proof system Prop if we can find a reduction function f which maps structures $\mathfrak{A}$ to inputs $f(\mathfrak{A})$ for Prop such that $\mathfrak{A} \in \mathcal{K}$ if, and only if, Prop can prove that $f(\mathfrak{A})$ is not satisfiable. It is clear that we only want to allow *simple* reduction functions f, because otherwise the computation of the encoding could already contain part of the work to solve the problem. Coming from the area of finite model theory the obvious and natural formalisation for "f being simple" is to say that f is definable in *first-order logic* (FO). We introduce the precise technical definition of such reductions, which is the notion of a *first-order interpretation*, in Section 2. Note that for the two examples we discussed above the encoding functions are FO-definable.

Having established this definition it turns out that our classification problem is really about understanding the expressive power of the *Lindström extensions* of first-order logic by *generalised quantifiers* for propositional proof systems Prop. We denote these logics by FO(Prop). The basic idea of the logic FO(Prop) is just to extend first-order logic by new quantifiers $\mathcal{Q}_{\text{Prop}}$ which are capable of simulating Prop. In other words, we just incorporate into first-order logic the power to simulate Prop in an explicit way. Note that the logics FO(Prop) are really nothing more than a formalisation of the concept of oracle Turing-machines with access to Prop in the world of first-order logic (the oracle calls to the proof system Prop correspond to applications of the new generalised quantifiers). Again, the precise technical definitions of the Lindström extensions FO(Prop) can be found in Section 2. Having defined these logics, we can now say that a problem $\mathcal{K}$ can be solved in a proof system Prop if, and only if, it is definable in FO(Prop). For instance, we saw that $\mathcal{K}_{\text{Reach}} \in \text{FO}(\text{Horn-Res}) \cap \text{FO}(2\text{-Res})$.

We are prepared to state our main results in a formal way. We first look at the restrictions of resolution we mentioned before, Horn-Res and k-Res, for $k \geq 2$. It turns out that Horn-Res can solve precisely those problems which are definable in least-fixed point logic (LFP), that is FO(Horn-Res) = LFP. This follows by the well-known result that the problem of computing winning positions in reachability games (known as GAME or alternating reachability) is complete for LFP with respect to FO-reductions. More interestingly, we proceed to show that k-Res, for every $k \geq 2$, is less powerful than Horn-Res. In fact, FO(2-Res) = FO(TC), where FO(TC) is the extension of first-order logic by a transitive closure operator. Moreover, we prove that, for every $k \geq 3$, FO(k-Res) = EFP, where EFP is the *existential* fragment of least fixed-point logic which is known to be a strict fragment of full least fixed-point logic. One can also show that the Lindström extensions for Horn resolution and width-k resolution have different structural properties. While for FO(Horn-Res) a single application of a $\mathcal{Q}_{\text{Horn-Res}}$ quantifier suffices to obtain the full expressive power, nesting of $\mathcal{Q}_{k\text{-Res}}$ quantifiers is needed for the logics FO(k-Res). For lack of space, details will be deferred to the full version of this paper.

We then turn our attention to the monomial variant of the polynomial calculus (mon-PC), which is a proof system based on algebraic reasoning techniques. Its restriction to polynomials of degree at most k, denoted by mon-PC$_k$, gives us an interesting polynomial-time proof system which is known to be much stronger than bounded-width resolution and Horn resolution. Accordingly, we can prove that the logic FO$^+$(mon-PC$_k$) is more powerful than all logics based on restrictions of resolution that we considered before. In fact, we show that FO$^+$(mon-PC$_k$), for $k \geq 2$, has the same expressive power as fixed-point logic with counting (FPC) which is a very expressive logic well-studied in descriptive complexity theory [15, 23] (here, FO$^+$ denotes the extension of FO by a numeric sort to match the setting of FPC).

Finally, we discuss applications of our model-theoretic characterisations of propositional proof systems. For instance, we can make statements about computational problems with regards to their solvability in one of these proof systems by using known (un-)definability results for fixed-point logics. Furthermore, we show how one can apply our logical characterisations of proof systems in order to transfer lower bounds from finite model theory to propositional proof complexity. In particular, we can easily reprove many lower bounds on the resolution sizes and widths for various families of propositional formulas.

Related work. Let us discuss some related work. The most relevant result to mention here is the elegant characterisation by Atserias and Dalmau of resolution width in terms of the number of pebbles required to win an existential pebble game played on a given CNF-formula and a structural encoding of truth assignments [2, 4]. This somehow resembles our result that bounded width resolution corresponds to *existential* least fixed-point logic. Using their game-theoretic characterisation, Atserias and Dalmau can reprove many of the known lower bounds on resolution width. Again, this is similar to the applications we give in Section 5.1.

However, what makes our setting different from the approach of Atserias and Dalmau is that we always consider the power of proof systems only *up to logical reductions*. This reflects, for example, in our result saying that FO(3-Res) = FO(4-Res), i.e. that 3-Res has the same expressive power as 4-Res. But, certainly, this only holds if we allow first-order reductions to transform inputs between 4-Res and 3-Res. Hence, we obtain a much less precise characterisation of resolution width. However, we think that the main advantage of our approach is its robustness. For instance, in the situation of lower bound proofs, we can avoid playing pebble games directly on the inputs to proof systems, such as CNF-formulas, but instead it suffices to play suitable games on pairs of structures in which these inputs interpret. This can make the description of winning strategies much simpler. Furthermore, our setting allows us to prove lower bound results much more independently from a concrete encoding of a problem, because of the fact that our logics are closed under logical interpretations, cf. discussion on the graph isomorphism problem in Section 5.1. Indeed, we can obtain lower bounds not only for one concrete family of inputs, but for any other family to which this family reduces to. For example, it is easy to see that our arguments in Corollary 20 about the lower bound for the 3-colourability problem actually go through for *any* other family of k-CNF-formulas to which one can reduce, in first-order logic say, the problem of solving a linear equation system over the two-element field (in which each equation has at most three variables) by using k-CNF-formulas with linearly many propositional variables.

Besides this, we want to mention the series of papers [5, 9, 22, 21] which establish surprisingly tight connections between the equivalence of graphs in counting logic and their indistinguishability by linear programming techniques (Sherali-Adams relaxiations of graph isomorphism polytopes) and algebraic propositional proof system. Similar to our applications, these results also allow the transfer of lower bounds from finite model theory to get lower bounds on proof complexity. In particular, we use notions and ideas of [9] in Section 4.

2 Preliminaries

Logical interpretations and Lindström quantifiers. Let $\mathcal{L}$ be a logic and σ, τ be signatures with $\tau = \{S_1, ..., S_\ell\}$. Let s_i denote the arity of S_i. An $\mathcal{L}[\sigma, \tau]$-*interpretation* is a tuple

$$I(\overline{z}) = (\varphi_\delta(\overline{x}, \overline{z}), \varphi_\approx(\overline{x}_1, \overline{x}_2, \overline{z}), \varphi_{S_1}(\overline{x}_1, ..., \overline{x}_{s_1}, \overline{z}), ..., \varphi_{S_\ell}(\overline{x}_1, ..., \overline{x}_{s_\ell}, \overline{z}))$$

where $\varphi_\delta, \varphi_\approx, \varphi_{S_1}, ..., \varphi_{S_\ell} \in \mathcal{L}[\sigma]$ and $\overline{x}, \overline{x}_1, ..., \overline{x}_{s_\ell}$ are tuples of pairwise distinct variables of the same length d and $\overline{z}$ is a tuple of variables pairwise distinct from the x-variables. We call d the *dimension* and $\overline{z}$ the *parameters* of $\mathcal{I}(\overline{z})$.

A d-dimensional $\mathcal{L}[\sigma,\tau]$-interpretation $\mathcal{I}(\overline{z})$ defines a partial mapping $\mathcal{I} : Str(\sigma,\overline{z}) \to Str(\tau)$ in the following way: For $(\mathfrak{A}, \overline{z} \mapsto \overline{a}) \in Str(\sigma,\overline{z})$ we obtain a τ-structure $\mathfrak{B}$ over the universe $\{\overline{b} \in A^d \mid \mathfrak{A} \models \varphi_\delta(\overline{b},\overline{a})\}$, setting $S_i^{\mathfrak{B}} = \{(\overline{b}_1,..,\overline{b}_{s_i}) \in B^{s_i} \mid \mathfrak{A} \models \varphi_{S_i}(\overline{b}_1,...,\overline{b}_{s_i},\overline{a})\}$ for each $S_i \in \tau$. Moreover let $\mathcal{E} = \{(\overline{b}_1,\overline{b}_2) \in A^d \times A^d \mid \mathfrak{A} \models \varphi_\approx(\overline{b}_1,\overline{b}_2,\overline{a})\}$. Now we define

$$\mathcal{I}(\mathfrak{A}, \overline{z} \mapsto \overline{a}) := \begin{cases} \mathfrak{B}/\mathcal{E} & \text{if } \mathcal{E} \text{ is a congruence relation on } \mathfrak{B} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We say that $\mathcal{I}$ interprets $\mathfrak{B}/\mathcal{E}$ in $\mathfrak{A}$.

Let $\mathcal{L}$ be a logic and $\mathcal{K} \subseteq Str(\tau)$ a class of τ-structures with $\tau = \{S_1,...,S_\ell\}$. The *Lindström extension* $\mathcal{L}(\mathcal{Q}_\mathcal{K})$ of $\mathcal{L}$ by *Lindström quantifiers* for the class $\mathcal{K}$ is obtained by extending the syntax of $\mathcal{L}$ by the following formula creation rule:

> Let $\varphi_\delta, \varphi_\approx, \varphi_{S_1}, ..., \varphi_{S_\ell}$ be formulas in $\mathcal{L}(\mathcal{Q}_\mathcal{K})$ that form an $\mathcal{L}[\sigma,\tau]$-interpretation $\mathcal{I}(\overline{z})$. Then $\psi(\overline{z}) = \mathcal{Q}_\mathcal{K}\mathcal{I}(\overline{z})$ is a formula in $\mathcal{L}(\mathcal{Q}_\mathcal{K})$ over the signature σ, with $(\mathfrak{A}, \overline{z} \mapsto \overline{a}) \models \mathcal{Q}_\mathcal{K}\mathcal{I}(\overline{z})$, if, and only if, $\mathfrak{B} := \mathcal{I}(\mathfrak{A}, \overline{z} \mapsto \overline{a})$ is defined and $\mathfrak{B} \in \mathcal{K}$.

Fixed-point logic with counting. We assume that the reader is familiar with least fixed-point logic, denoted LFP. In finite model theory, a very important extension of LFPis fixed-point logic with counting, FPC. FPC is evaluated on *two-sorted structures*. For any finite, one-sorted σ-structure $\mathfrak{A}$ with universe A, we define the two-sorted extension $\mathfrak{A}^+ := \mathfrak{A} \uplus (\{0,...,|A|\};<)$, where $<$ is the canonical ordering on $\{0,...,|A|\}$. We call the thus extended vocabulary σ^+. The elements of A form the *point sort* and $\{0,...,|A|\}$ is called the *numeric sort*. Fixed-point logic with counting (FPC) is the extension of least-fixed point logic over such two-sorted structure by counting quantifiers, so that we have formulas $\exists^{\geq\lambda} x\varphi$, where λ is a numeric variable, saying that there exist at least λ many points $a \in A$ making $\varphi(a)$ true. The importance of FPC comes from the fact that it can express many fundamental algorithmic techniques and comes very close to being a logic for polynomial time. For more details on FPC, we refer to [15, 24].

Representing propositional formulas as relational structures. Propositional formulas (in CNF) can be represented as structures of some fixed vocabulary in several ways. We shall briefly discuss two possibilities. Since these, and others, are mutually interpretable into each other by simple formulas, it does not really matter which representation we choose; the corresponding Lindström extensions of FO will all have the same expressive power. Perhaps the most obvious representation of a CNF-formula ψ as a structure $\mathfrak{A}(\psi)$ is based on the vocabulary $\{C, V, P, N\}$; the universe of $\mathfrak{A}(\psi)$ consists of the variables and the clauses of ψ, the monadic relations V and C identify the variables and clauses, respectively, and the binary relations P and N specify which variables appear positively and negatively in which clauses; so Pvc is true in $\mathfrak{A}(\psi)$ if the variable v appears positively in the clause c, and analogously for N. A different representation, that sometimes leads to more elegant logical descriptions works with the set L of literals and with a self-inverse bijection $\neg : L \to L$, so that ψ would be represented by $\mathfrak{A}(\psi) = (A, C, L, \neg, \in)$ where A is the set of clauses and literals, $\neg(x)$ is the complementary literal to x, and $x \in c$ means that the literal x occurs in the clause c.

3 Resolution and Fixed-Point Logics

3.1 Horn Resolution captures Least Fixed-Point Logic

Let posLFP be the fragment of LFP-formulas that are in negation normal form (i.e. negation is applied only to input atoms), in which each fixed-point variable is bound only once, and that do not make use of greatest fixed points. Further, let EFP_0 be the basic existential fragment of LFP; it consists of those formulas in posLFP whose quantifiers are all existential.

It is known that, on finite structures (but not in general), every LFP-formula can be effectively translated into an equivalent one in posLFP. On the other side EFP_0 is strictly weaker; it has the same expressive power as Datalog with negation of input atoms.

▶ **Theorem 1.** *For every $\varphi \in \mathrm{posLFP}[\tau]$ there is a first-order interpretation I_φ that maps finite τ-structures to propositional Horn formulas $\psi_{\mathfrak{A},\varphi}$ such that $\mathfrak{A} \models \varphi$ if, and only if, $\psi_{\mathfrak{A},\varphi}$ is unsatisfiable. Further, if φ is in EFP_0 then all clauses in $\psi_{\mathfrak{A},\varphi}$ have width at most three.*

Proof. Fix a formula $\varphi \in \mathrm{posLFP}[\tau]$. For every finite τ-structure $\mathfrak{A}$, with universe A, we construct the propositional Horn formula $\psi_{\mathfrak{A},\varphi}$ as follows. An *instantiated subformula* of φ is an expression $\beta(\overline{a})$ which is obtained by taking some subformula $\beta(\overline{x})$ of φ and by instantiating every free variable x by some element $a \in A$. We now take for every instantiated subformula β of φ a propositional variable X_β, and inductively define a set $C(\mathfrak{A},\varphi)$ of clauses as follows.

1. If β is a τ-literal then we add $1 \to X_\beta$ in case that $\mathfrak{A} \models \beta$ and $X_\beta \to 0$ in case $\mathfrak{A} \not\models \beta$.
2. If $\beta = \eta \vee \vartheta$ we add the clauses $X_\eta \to X_\beta$ and $X_\vartheta \to X_\beta$.
3. If $\beta = \eta \wedge \vartheta$ we add the clause $X_\eta \wedge X_\vartheta \to X_\beta$.
4. If $\beta = \exists x \eta(x)$ then we add all clauses $X_{\eta(a)} \to X_\beta$ for $a \in A$.
5. If $\beta = \forall x \eta(x)$ then we add the clause $(\bigwedge_{a\in A} X_{\eta(a)}) \to X_\beta$.
6. If $\beta = [\mathbf{lfp}\, Rx\,.\,\eta](\overline{a})$ or $\beta = R\overline{a}$, then we add the clause $X_{\eta(\overline{a})} \to X_\beta$.

By induction, it readily follows that the minimal model of all these clauses sets the variable X_β to true if, and only if $\mathfrak{A} \models \beta$ (with fixed-point variables interpreted by their least fixed-point on $\mathfrak{A}$). Let now $\psi_{\mathfrak{A},\varphi}$ be defined as the conjunction of all clauses in $C(\mathfrak{A},\varphi)$ together with $X_\varphi \to 0$. Then $\psi_{\mathfrak{A},\varphi}$ is unsatisfiable if, and only if, $\mathfrak{A} \models \varphi$.

We observe that the only clauses of size larger than three are those coming from universal quantifiers. Hence, if there are no universal quantifiers, the formula only has clauses of size at most three. Finally it is clear that, for every fixed $\varphi \in \mathrm{posLFP}[\tau]$, we can interpret (a representation of) the formula $\psi(\mathfrak{A},\varphi)$ inside $\mathfrak{A}$, by using an FO-interpretation I_φ. ◀

This shows that $\mathrm{LFP} \leq \mathrm{FO}(\textsc{Horn-Res})$. Actually we established a stronger result.

▶ **Theorem 2.** *For every formula $\varphi \in \mathrm{LFP}$ there exists a first-order interpretation J_φ such that $\mathcal{Q}_{\textsc{Horn-Res}}(J_\varphi)$ is equivalent to φ on finite structures. In particular, each* LFP*-formula can be translated into an equivalent* FO(Horn-Res)*-formula with a single application of the generalised quantifier $\mathcal{Q}_{\textsc{Horn-Res}}$.*

We are ready to prove that FO(Horn-Res) has the same expressive power as LFP.

▶ **Theorem 3.** *On finite structures,* $\mathrm{LFP} = \mathrm{FO}(\textsc{Horn-Res})$.

It remains to show that $\mathrm{FO}(\textsc{Horn-Res}) \leq \mathrm{LFP}$, that is we have to express Horn resolution in LFP. Recall that a propositional Horn formula ψ admits a derivation of the empty clause if, and only if, ψ contains a clause in which all variables appear negatively,

written $X_1 \dots X_k \to 0$, such that all unit clauses $\{X_i\}$ for $i = 1, \dots, k$ can be derived from ψ by Horn resolution.

Let ψ be presented as a structure $\mathfrak{A}(\psi)$ with universe $C \cup V$ and vocabulary $\{C, V, P, N\}$. Let D be the set of variables $v \in V$ such that the clause $\{v\}$ can be derived from ψ by Horn resolution. Then ψ is unsatisfiable if, and only if, $\mathfrak{A}(\psi) \models \exists c(Cc \wedge \neg\exists x Pxc \wedge \forall x(Nxc \to Dx))$. The set D is definable by the LFP-formula $[\mathbf{lfp}\, Dx\,.\, \exists c(Pxc \wedge \forall y(Nyc \to Dy))](x)$.

3.2 Bounded Width Resolution and Existential Least Fixed-Point Logic

Intuitively, *existential least fixed-point logic* (EFP) extends EFP_0 by stratified negation. This means that it permits fixed-point formulas over existential formulas which may depend on closed fixed-point relations, defined in a lower stratum, and these can be used also in negated form. Thus, negation (and hence, implicitly, also universal quantifiers) are present in a limited form, but least fixed-point recursions may never go through negation or universal quantification. In fact, EFP is equivalent to Stratified Datalog.

▶ **Definition 4.** Existential fixed-point logics $\mathrm{EFP} := \bigcup_{\ell \geq 0} \mathrm{EFP}_\ell$ generalises EFP_0 as follows. The stratum $\mathrm{EFP}_{\ell+1}$ is the closure under disjunction, conjunction and existential quantification of formulas of form $[\mathbf{lfp}\ R\overline{x}.\exists\overline{y}\varphi(R, \overline{x}, \overline{y})](\overline{x})$ where $\varphi(R, \overline{x}, \overline{y})$ is obtained from a quantifier-free formula, that may contain positive and negative occurrences of additional relations $S_1, \dots, S_m$, by substituting these relations by formulas from EFP_ℓ.

Notice that first-order logic FO is contained in EFP, but not in any bounded level EFP_ℓ, because every quantifier alternation in FO must be simulated by an additional level of stratified negation. For the same reason EFP, but none of its levels EFP_ℓ, is closed under first-order operations. As a consequence of Theorem 1 we can infer

▶ **Theorem 5.** *On finite structures,* $\mathrm{EFP} \leq \mathrm{FO}(3\text{-Res})$.

Proof. Theorem 1 directly establishes this for EFP_0. So assume that the claim is established for EFP_ℓ. Every formula in $\mathrm{EFP}_{\ell+1}$ can be written as an EFP_0-formula over predicates that are EFP_ℓ-definable. Hence, by applying Theorem 1 once more, it can be rewritten as an $\mathrm{FO}(3\text{-Res})$-formula over predicates that are themselves definable in $\mathrm{FO}(3\text{-Res})$. Since Lindström extensions of FO are closed under nesting of generalised quantifiers, it follows that also $\mathrm{EFP}_{\ell+1} \leq \mathrm{FO}(3\text{-Res})$. ◀

We require clauses of width 3 for translating EFP-formulas into Horn formulas. In fact, if we restrict to clauses of width 2, then we obtain the power of first-order logic with a transitive closure operator $\mathrm{FO}(\mathrm{TC})$. This immediately follows from the fact that satisfiability of 2-CNF formulas reduces to graph reachability, and from the reduction of graph reachability to the non-satisfiability problem for a 2-CNF formula that we described in the introduction.

▶ **Theorem 6.** *It holds that* $\mathrm{FO}(2\text{-Res}) = \mathrm{FO}(\mathrm{TC})$.

Simulating bounded width resolution in EFP. To describe resolution of width k, for any fixed $k \geq 1$, in EFP, we shall use the representation of CNF-formula ψ by structures $\mathfrak{A}(\psi) = (A, C, L, \neg, \in)$ where C is the set of clauses and L is the set of literals, and the universe is $A = C \cup L \cup \{0\}$. Further we shall describe the set of all derivable clauses of size at most k as a k-ary relation $D \subseteq (L \cup \{0\})^k$, that contains those k-tuples $(x_1, \dots, x_k)$ for which $\{x_i : i \leq k, x_i \neq 0\}$ is a clause that is derivable from ψ. This relation D is defined by a fixed-point formula $[\mathbf{lfp}\, D\overline{x}\,.\,\varphi(D, \overline{x})](\overline{x})$ where $\varphi(D, \overline{x})$ expresses the following. Either

1. there exists a clause $c \in C$ such that $c = \{x_1, \dots, x_k\} \setminus \{0\}$, or
2. there exist tuples $\overline{y}, \overline{z} \in D$ such that, for some i, j, the literal z_j is the negated literal to y_i, and $(\{y_1, \dots, y_k\} \cup \{z_1, \dots, z_k\}) \setminus \{y_i, z_j, 0\} = \{x_1, \dots, x_k\} \setminus \{0\}$.

When spelling out these equations in first-order logic, we can express $\varphi(\overline{x}, D)$ by an existential FO-formula $\exists \overline{y}\, \alpha(\overline{x}, \overline{y}, D, Q)$ where Q is FO-definable by a formula (with quantifier prefix $\exists^*\forall$) that does not depend on D. This yields a formula in EFP_1. Since EFP is closed under FO-operations, this proves

▶ **Theorem 7.** *On finite structures,* $\mathrm{FO}(k\text{-}\textsc{Res}) \leq \mathrm{EFP}$ *for all* $k \in \mathbb{N}$.

4 The Monomial-PC and Fixed-Point Logic with Counting

We turn our attention to the *monomial-PC* (MON-PC). The monomial-PC is a propositional proof system that is based on algebraic reasoning techniques and which lies between the *Nullstellensatz* proof system and the *polynomial calculus* (PC). It was introduced by Berkholz and Grohe in [9] in order to characterise the power of an important graph isomorphism test, the Weisfeiler-Lehman method, in terms of propositional proof complexity. We show in this section that the monomial-PC has precisely the same expressive power as fixed-point logic with counting (FPC), which is a natural and powerful logic of great importance in the area of descriptive complexity theory.

4.1 The Monomial-PC

We start with background on the polynomial calculus and its variant, the monomial-PC. Both systems refute the solvability of a given set of *(multivariate) polynomial equations* over some field $\mathbb{F}$ using proof rules that manipulate such equations. In this paper, $\mathbb{F}$ will always be the field of rationals $\mathbb{Q}$. We denote by $\mathbb{Q}[\vec{X}]$ the ring of polynomials in variables X_j, $j \in J$, for some (unordered) index set J and with coefficients in $\mathbb{Q}$. For a multi-index $\alpha : J \to \mathbb{N}$ we let the *monomial* X^α be defined as $X^\alpha = \Pi_{j \in J} X_j^{\alpha(j)}$. Then polynomials $f \in \mathbb{Q}[\vec{X}]$ can be written as $f = \sum_\alpha f_\alpha \cdot X^\alpha$ where the $f_\alpha \in \mathbb{Q}$ are coefficients from the field $\mathbb{Q}$ and such that $f_\alpha \neq 0$ for finitely many α only. The *degree* $\deg(X^\alpha)$ of a monomial X^α is defined as $|\alpha| = \sum_{j \in J} \alpha(j)$, and the degree of a polynomial is defined as the maximal degree of its monomials. A *polynomial equation* is an equation of the form $f = 0$ for a polynomial $f \in \mathbb{Q}[\vec{X}]$. For better readability, we usually omit the equality "$= 0$" when we specify polynomial equations, that is we identify polynomials $f \in \mathbb{Q}[\vec{X}]$ with the corresponding normalised polynomial equations $f = 0$. A *system of polynomial equations* is a set $\mathcal{P} = \{f_i : i \in I\}$ consisting of polynomials $f_i \in \mathbb{Q}[\vec{X}]$ for all $i \in I$ where I is an (unordered) index set. A *solution* of $\mathcal{P}$ is a common zero $\overline{a} \in \mathbb{Q}^J$ of all polynomials in $\mathcal{P}$. In what follows, we only consider systems $\mathcal{P} = \{f_i : i \in I\}$ which contain for every variable $X = X_j$, $j \in J$, the polynomial equation $(X^2 - X) = 0$. The axioms $(X^2 - X) = 0$ enforce that each variable $X = X_j$, $j \in J$, can only take values 0 or 1.

The polynomial calculus is based on the following result from algebra which is known as *Hilbert's Nullstellensatz*. It says that the non-solvability of the system $\mathcal{P} = \{f_i : i \in I\}$ is equivalent to the existence of polynomials $g_i \in \mathbb{Q}[\vec{X}], i \in I$, such that $\sum_{i \in I} g_i \cdot f_i = 1$. The polynomials g_i are called a *Nullstellensatz refutation* for the system $\mathcal{P}$. The idea of the polynomial calculus is to search for such polynomials g_i in a sequential way.

▶ **Definition 8.** The axioms and inference rules of *polynomial calculus* (PC) are as follows.

$$\frac{p \in \mathcal{P}}{p}\ \text{(Axioms)} \qquad \frac{f}{Xf}\ \text{(Multiplication)} \qquad \frac{g \quad f}{ag + bf}\ \text{(Linear combinations)}$$

The *monomial-PC* (MON-PC) is the restriction that permits the use of the multiplication rule only in the cases where f is either a monomial or the product of a monomial and an axiom. A polynomial equation system system $\mathcal{P}$ has a *refutation* of degree $k \geq 1$ in PC (or MON-PC) if the polynomial $1 \in \mathbb{Q}[\vec{X}]$ can be derived from $\mathcal{P}$ using the above rules using polynomials of degree at most k.

The polynomial calculus, and also the monomial-PC, are sound and, by Hilbert's Nullstellensatz, complete proof systems. However, this only holds if we do not restrict the degree of the polynomials which are allowed to occur in a refutation. In fact, the "degree of polynomials" for the PC (MON-PC) is a complexity measure which has similar properties as the "width of clauses" measure that we studied for the resolution proof system. If we restrict the PC (MON-PC) to polynomials of degree at most k, for some fixed $k \geq 1$, then the systems become incomplete, but admit proof search in polynomial time. In what follows, whenever we speak of the monomial-PC, we implicitly refer to the variant where we restricted the degree of polynomials to some constant $k \geq 1$. If we want to make this constant precise, then we denote the corresponding proof system by MON-PC$_k$. Another fact which we use throughout this section is that the axioms $(X^2 - X)$ guarantee that in (monomial-)PC proofs we can restrict ourselves to *multilinear* polynomials. To see this, say that we were able to derive the polynomial $p = X^2Y + Z$ within some (monomial)-PC proof. Of course, p is not multilinear. However, we can use the axiom $(X^2 - X)$ together with the "linear combination"-rule to reduce this polynomial to the corresponding multilinear polynomial $p' = XY + Z$. Indeed, $p' = p - Y(X^2 - X)$. Hence, restricting to multlinear polynomials, and modifying the multiplication rule accordingly with implicit linearisation, does not change the power of the corresponding proof systems. For a polynomial $p \in \mathbb{Q}[\vec{X}]$ we denote its *multilinearisation* by MultLin(p).

4.2 Monomial-PC in Fixed-Point Logic with Counting

We show that FPC can simulate MON-PC$_k$. For this, we have to agree on an encoding of a set $\mathcal{P}$ of rational, multilinear polynomials as relational structures. Similar to our representation of CNF-formulas described in Section 2, a natural encoding can be based on a many-sorted structure $\mathfrak{A}_\mathcal{P}$ whose universe is partitioned into sets of polynomials, (multilinear) monomials, variables, and rational coefficients that occur in $\mathcal{P}$. As usual, we represent rationals as fractions of integers using binary encoding. Hence, $\mathfrak{A}_\mathcal{P}$ should also provide a linear order of sufficient length to encode these binary strings. Again, the exact technical details are not important, as long as the encoding has some natural properties, such as FO-definability of the class of valid encodings. By a slight abuse of notation, we also denote by MON-PC$_k$ the class of structures $\mathfrak{A}_\mathcal{P}$ which encode a system $\mathcal{P}$ which can be refuted in MON-PC$_k$.

▶ **Theorem 9.** *For every $k \geq 1$,* MON-PC$_k \in$ FPC.

Given a set of multilinear polynomials $\mathcal{P}$ of degree at most k, we consider the set $V_\mathcal{P}$ of multilinear polynomials which can be derived from $\mathcal{P}$ within MON-PC$_k$. The first observation is that $V_\mathcal{P}$ is a $\mathbb{Q}$-linear space. Since $V_\mathcal{P}$ only contains multilinear polynomials of degree at most k, we can naturally associate polynomials $p \in V_\mathcal{P}$ with vectors $p \in \mathbb{Q}^{M_k}$ where the index set M_k denotes the set of all multilinear monomials of degree at most k. For fixed $k \geq 1$, this set M_k is of polynomial size.

To prove Theorem 9 we express in FPC an inductive algorithm (based on a similar algorithm for the full polynomial calculus in [13]) for computing a generating set of the space $V_\mathcal{P}$. Then, in order to see whether MON-PC$_k$ can refute the system $\mathcal{P}$, we simply have to check whether the constant polynomial 1 is contained in $V_\mathcal{P}$, see Figure 1.

Input: Set of multilinear polynomials $\mathcal{P} \subseteq \mathbb{Q}^{M_k}$
Output: $\mathcal{B} \subseteq \mathbb{Q}^{M_k}$ such that $\langle \mathcal{B} \rangle = V_{\mathcal{P}}$.
Initialisation (lift all axioms in $\mathcal{P}$)
$\mathcal{B} := \{\text{MultLin}(m \cdot p) \mid p \in \mathcal{P}, m \text{ a monomial s.th. } \deg(\text{MultLin}(m \cdot p)) \leq k\}$
repeat
  for all monomials $m \in \langle \mathcal{B} \rangle$, $\deg(m) < k$ **do**
    $\mathcal{B} := \mathcal{B} \cup \{\text{MultLin}(X \cdot m) : \text{for some variable } X\}$
  end for
until $\mathcal{B}$ remains unchanged
return $\mathcal{B}$

Figure 1 FPC-procedure to define generating set for $V_{\mathcal{P}}$.

During the run of the algorithm we have that $\langle \mathcal{B} \rangle \leq V_{\mathcal{P}}$. Moreover, after termination it holds that $\langle \mathcal{B} \rangle = V_{\mathcal{P}}$. We further observe that after the initialisation step we only add *monomials* to the set $\mathcal{B}$. Since there are only polynomially many different monomials of degree at most k, for a fixed k, this means that the algorithm is guaranteed to terminate after a polynomial number of iterations.

It is not obvious how to express this algorithm in FPC. Most steps, such as the representation of the set $\mathcal{B}$ and the multilinearisation of polynomials, are easy to formalise, but there is a severe obstacle hidden in the condition for the main loop. Here, we want to iterate, in parallel, through all monomials $m \in \langle \mathcal{B} \rangle$. This condition "$m \in \langle \mathcal{B} \rangle$" translates to solving a linear equation system over $\mathbb{Q}$. Although it is provably impossible to express the method of Gaussian elimination in FPC, since it requires arbitrary choices during its computation, and although FPC cannot define the solvability of linear equation systems over finite fields [3], it is known [17] that FPC can indeed express solvability of linear equation systems over the rationals.

▶ **Theorem 10** ([17]). *The solvability of linear equation systems over $\mathbb{Q}$ is definable in* FPC.

Using this result we can express the algorithm in FPC. In order to complete our proof of Theorem 9 we recall that MON-PC$_k$ can refute $\mathcal{P}$ if, and only if, $1 \in \langle \mathcal{B} \rangle = V_{\mathcal{P}}$. This last assertion, again, reduces to a linear equation system over $\mathbb{Q}$ and can thus be defined in FPC.

4.3 Monomial-PC captures Fixed-Point Logic with Counting

Next we show that the monomial-PC can simulate fixed-point logic with counting. We first observe, however, that the logic FO(MON-PC$_k$) does *not* suffice for this purpose. This is due to the fact that FPC has access to the second numeric sort, on which it can perform arbitrary polynomial time computations, whereas FO(MON-PC$_k$) is evaluated over standard single sorted input structures. To overcome this mismatch we have to extend the logic FO(MON-PC$_k$) to the second-sorted framework as well. We denote this extension of FO(MON-PC$_k$) by FO$^+$(MON-PC$_k$). As in the case of FPC, this means that formulas are evaluated over extensions $\mathfrak{A}^+$ of relational structures $\mathfrak{A}$ by a second numeric sort, as defined in Section 2. In particular, interpretations for the Lindström quantifiers can make use of the second numeric sort, and we require this capability in the proof of our following result.

▶ **Theorem 11.** *For every $k \geq 2$,* FPC $\leq$ FO$^+$(MON-PC$_k$).

An elegant way to prove Theorem 11 is to use a game-theoretic characterisation of FPC which was recently established in [20]. It is based on the notion of so called *threshold games*.

A *threshold game* is a two-player game played on a directed graph $G = (V, E)$ that is equipped with a *threshold function* $\vartheta\colon V \to \mathbb{N}$. This function satisfies that $\vartheta(v) \leq \delta(v) + 1$ for all $v \in V$, where $\delta(v)$ denotes the out-degree of v in G. Moreover, there is a designated vertex $s \in V$ at which each play starts. A play is a sequence of G-nodes that arises according to the following rules. At the current position $v \in V$, Player 0 first selects a set $X \subseteq vE = \{w : (v, w) \in E\}$ with $|X| \geq \vartheta(v)$. Then Player 1 chooses a node $w \in X$ and the play moves on to w. A player who cannot move loses. Hence Player 0 wins at all nodes in $T_0 := \{v \in V \mid \vartheta(v) = 0\}$ and Player 1 at all nodes in $T_1 := \{v \in V \mid \delta(v) < \vartheta(v)\}$.

In [20] it is shown that threshold games provide appropriate model-checking games $\mathcal{T}(\mathfrak{A}, \varphi)$ for any finite structure $\mathfrak{A}$ and any formula $\varphi \in \mathrm{FPC}$. Since fixed-point evaluations on finite structures can be uniformly unraveled to first-order evaluations, we can in fact assume that the game graphs of these threshold games are acyclic. For any fixed FPC-formula φ, these model checking games are polynomially bounded in the size of the input structure and can, in fact, be interpreted in (two-sorted) input structures using a first-order interpretation. This is related to the transformation of FPC-formulas into uniform families of polynomial-size threshold circuits, as used for instance in [23] and [1].

▶ **Theorem 12** ([20]). *For every* FPC*-formula φ there is a first-order interpretation I_φ which, for every finite structure $\mathfrak{A}$, interprets in $\mathfrak{A}^+$ an acyclic threshold game $\mathcal{G}(\mathfrak{A}, \varphi)$ such that $\mathfrak{A} \models \varphi$ if, and only if, Player 0 has a winning strategy for $\mathcal{G}(\mathfrak{A}, \varphi)$.*

It remains to show that the monomial-PC can define winning regions in acyclic threshold games. Given an acyclic threshold game $\mathcal{G} = (G = (V, E), \vartheta)$, we construct an axiom system $\mathcal{P}(\mathcal{G})$ which consists of polynomial equations of degree at most two. For every node $v \in V$ in the threshold game $\mathcal{G}$, the system $\mathcal{P}(\mathcal{G})$ contains a variable X_v. Let us denote by $W_\sigma^{\mathcal{G}}$ the winning region of Player σ in $\mathcal{G}$. Then $\mathcal{P}(\mathcal{G})$ satisfies the following:

- if $v \in W_0^{\mathcal{G}}$, then $X_v = 1$ is derivable from $\mathcal{P}(\mathcal{G})$ in MON-PC_2;
- if $v \in W_1^{\mathcal{G}}$, then $X_v = 0$ is derivable from $\mathcal{P}(\mathcal{G})$ in MON-PC_2;
- $\mathcal{P}(\mathcal{G})$ is consistent; in particular, either $X_v = 1$ or $X_v = 0$ is derivable for every $v \in V$;

If we can construct such a system $\mathcal{P}(\mathcal{G})$ via an FO-interpretation in $\mathcal{G}$, then this completes our proof of Theorem 11. In fact, it then follows that FO^+(MON-PC_2) can define winning regions in acyclic threshold games: a node $v \in V$ is in the winning region of Player 0 if, and only if, the system $\mathcal{P}(\mathcal{G}) \cup \{X_v = 0\}$ can be refuted in MON-PC_2.

Recall that $vE = \{w \in V : (v, w) \in E\}$, for $v \in V$, denotes the set of successors of v. Further, we let $\mathrm{s}(v)$ denote the number of successors of v, and we let $\mathrm{ws}(v)$ denote the number of successors of v which are in the winning region of Player 0, that is $\mathrm{s}(v) = |vE|$ and $\mathrm{ws}(v) = |vE \cap W_0^{\mathcal{G}}|$. We denote the set of non-terminal positions by $\mathrm{NonTerm} = \{v \in V : \mathrm{s}(v) > 0\}$. The system $\mathcal{P}(\mathcal{G})$ uses the following set of variables:

- a variable X_v, for every $v \in V$,
- a variable Y_v^m for every $v \in \mathrm{NonTerm}$, and $0 \leq m \leq \mathrm{s}(v)$,
- a variable $Z_v^m[u \mapsto j]$ for every $v \in \mathrm{NonTerm}$, $1 \leq m \leq \mathrm{s}(v)$, $1 \leq j \leq m$, $u \in vE$.

The intuition is that the variables X_v encode the winning regions of both players, as described above. Moreover, the variables Y_v^m should indicate whether $\mathrm{ws}(v) = m$, in the following way: if $\mathrm{ws}(v) \neq m$, then $Y_v^m = 0$ is derivable, and if $\mathrm{ws}(v) = m$, then $Y_v^m = 1$ is derivable. The variables $Z_v^m[u \mapsto j]$ are auxiliary variables used to encode this last condition, cf. [9]. The system $\mathcal{P}(\mathcal{G})$ consists of the following axioms:

(T) For $v \in T_0 : X_v = 1$ and for $v \in T_1 : X_v = 0$

(C) For $v \in \mathrm{NonTerm}, 1 \leq m \leq \mathrm{s}(v), u \in vE$: $\sum_{j=1}^{m} Z_v^m[u \mapsto j] - Y_v^m = 0$

For $v \in \mathrm{NonTerm}, 1 \leq m \leq \mathrm{s}(v), 1 \leq j \leq m$: $\sum_{u \in vE} X_u Z_v^m[u \mapsto j] - Y_v^m = 0$

For $v \in \mathrm{NonTerm}$: $\sum_{u \in vE} X_u \cdot Y_v^0 = 0$

(E) For $v \in V : (1 - X_v) - \sum_{m=0}^{\vartheta(v)-1} Y_v^m = 0$ and $X_v - \sum_{m=\vartheta(v)}^{\mathrm{s}(v)} Y_v^m = 0$

We also add for each variable $X = X_v$ a dual variable $\overline{X}$ with the axiom (N) $1 - X = \overline{X}$.

▶ **Lemma 13.** *The system $\mathcal{P}(\mathcal{G})$ is consistent.*

Proof. We define an intended model of $\mathcal{P}(\mathcal{G})$. For X-variables, we set $X_v := 1$, if $v \in W_0^{\mathcal{G}}$, and $X_v := 0$, if $v \in W_1^{\mathcal{G}}$. For Y-variables, we set $Y_v^m := 1$, if $\mathrm{ws}(v) = m$, and $Y_v^m := 0$ if $m \neq \mathrm{ws}(v)$. For Z-variables, we set $Z_v^m[u \mapsto j] := 0$ for all non-terminal positions $v \in V$, $u \in vE$, and $j \in \{1, \ldots, m\}$, if $m \neq \mathrm{ws}(v)$. For $m = \mathrm{ws}(v) > 0$, we let $vE \cap W_0^{\mathcal{G}} = \{u_1, \ldots, u_m\}$. We then set $Z_v^m[u_i \mapsto j] = 1$ if $j = i$, and $Z_v^m[u_i \mapsto j] = 0$ for $j \neq i$. Moreover, for $u \in vE \setminus W_0^{\mathcal{G}}$, we set $Z_v^m[u \mapsto 1] = 1$, and $Z_v^m[u \mapsto j] = 0$ for $j \in \{2, \ldots, m\}$. ◀

▶ **Lemma 14.** *If $v \in W_0^{\mathcal{G}}$, then we can derive $X_v = 1$ from $\mathcal{P}(\mathcal{G})$ in* MON-PC_2*; and if $v \in W_1^{\mathcal{G}}$, then $X_v = 0$ is derivable from $\mathcal{P}(\mathcal{G})$ in* MON-PC_2.

Proof. We start with a small remark. Assume that we can derive $(1 - X)$ for a variable $X = X_v$, $v \in V$. We show how to derive $V(1 - X)$ for every variable V. This is clearly possible in the full polynomial calculus. In the monomial-PC, however, we cannot multiply $(1 - X)$ by V, since $(1 - X)$ is neither a monomial nor an axiom. Instead, we use our negation axioms: We obtain $\overline{X} = 0$ by subtracting $X = 1$ from (N). Since (N) is an axiom, we can multiply it by V; also, $\overline{X}$ is a monomial and it can be multiplied by V. Thus, $V(1 - X - \overline{X}) + V\overline{X} = V(1 - X)$ can be derived. We make use of this in what follows.

The proof is by induction on the height of the subgame rooted at $v \in V$ (recall that $\mathcal{G}$ is acyclic). For terminal positions $v \in V$, the claim is obvious. Assume $v \in V$ is a non-terminal position. Let $W_0(v) = vE \cap W_0^{\mathcal{G}}$ and $W_1(v) = vE \cap W_1^{\mathcal{G}}$. By the induction hypothesis we know that we can derive in MON-PC_2 for every $u \in W_0(v)$ the equation $X_u = 1$ and for every $u \in W_1(v)$ the equation $X_u = 0$.

Let $m > 0$. Consider an equation of the form $\sum_{u \in vE} X_u Z_v^m[u \mapsto j] - Y_v^m = 0$ for $j \in \{1, \ldots, m\}$ of type (C). We have $vE = W_0(v) \uplus W_1(v)$. For every Z-variable and for every $u \in W_0(v)$ we can derive $ZX_u = Z$ in MON-PC_2, and for every $u \in W_1(v)$ we can derive $ZX_u = 0$ in MON-PC_2. Hence, we can simplify these equations of type (C) as $\sum_{u \in W_0(v)} Z_v^m[u \mapsto j] - Y_v^m = 0$ for $j \in \{1, \ldots, m\}$ in MON-PC_2.

Next, we consider for every $u \in W_0(v)$ the equations $\sum_{j=1}^{m} Z_v^m[u \mapsto j] - Y_v^m = 0$, again of type (C). We combine these two sets of equations as follows:

$$\sum_{j \in \{1,\ldots,m\}} \left(\sum_{u \in W_0(v)} Z_v^m[u \mapsto j] - Y_v^m \right) - \sum_{u \in W_0(v)} \left(\sum_{j=1}^{m} Z_v^m[u \mapsto j] - Y_v^m \right) = (m - \mathrm{ws}(v)) Y_v^m.$$

Hence, for every $m > 0$, $m \neq \mathrm{ws}(v)$, we can derive $Y_v^m = 0$ in MON-PC$_2$. Indeed, also in the case where $m = 0 < \mathrm{ws}(v)$ we can derive $Y_v^m = 0$. In this case we just use the equation $\sum_{u \in vE} X_u Y_v^0 = 0$. Using the same arguments as above, this equation simplifies to $\mathrm{ws}(v) \cdot Y_v^0 = 0$. Hence, if $\mathrm{ws}(v) > 0$, we can also derive $Y_v^0 = 0$. Note that the two equations of type (E) can be combined to the equation $\sum_{m=0}^{s(v)} Y_v^m = 1$. Hence, altogether we showed the following. For all $0 \leq m \leq s(v)$ it holds that:

- if $m = \mathrm{ws}(v)$, then we can derive $Y_v^m = 1$ in MON-PC$_2$; and
- if $m \neq \mathrm{ws}(v)$, then we can derive $Y_v^m = 0$ in MON-PC$_2$.

Having this, the claim follows immediately by using the equations of type (E). Finally, the system $\mathcal{P}(\mathcal{G})$ can easily be obtained from the game $\mathcal{G}$ by means of an FO-interpretation. ◀

In summary, we have seen that defining the winning regions in acyclic threshold games is an FPC-complete problem, with respect to FO$^+$-reductions, and that the winning regions in such games can be defined in FO$^+$(MON-PC$_2$). This completes the proof of Theorem 11 and, together with Theorem 9, establishes the main theorem of this section.

▶ **Theorem 15.** *For every $k \geq 2$,* FPC = FO$^+$(MON-PC$_k$).

5 Applications

5.1 Lower Bounds on Resolution Width and Size

Our characterisation of bounded width resolution in terms of EFP-definability reproves many lower bounds on the resolution size and width for families of propositional formulas.

We denote the finite-variable fragment of *infinitary logic* by $L^\omega_{\infty\omega}$, that is $L^\omega_{\infty\omega}$ is the finite-variable fragment of the extension of first-order logic by infinite conjunctions and disjunctions. Further, we denote by $L^\ell_{\infty\omega}$ the ℓ-variable fragment of $L^\omega_{\infty\omega}$; we have $L^\omega_{\infty\omega} = \bigcup_{\ell \geq 1} L^\ell_{\infty\omega}$. We write $\mathfrak{A} \equiv^\ell \mathfrak{B}$ if two structures $\mathfrak{A}$ and $\mathfrak{B}$ cannot be distinguished by any sentence of the ℓ-variable fragment $L^\ell_{\infty\omega}$ of $L^\omega_{\infty\omega}$. It is well-known that EFP, and even LFP, can be embedded into the logic $L^\omega_{\infty\omega}$. More precisely, if $\varphi \in$ LFP is a sentence with ℓ (first-order) variables, then we can find an equivalent sentence φ^* in $L^\ell_{\infty\omega}$. We formulate the following definition and result with respect to $L^\ell_{\infty\omega}$-interpretations, instead of FO-interpretations.

▶ **Definition 16.** Let $k \geq 1$ and let $(\Phi_n) = (\Phi_n)_{n \geq 1}$ be a family of k-CNF formulas. Moreover, let $\mathcal{I}$ be an $L^\ell_{\infty\omega}$-interpretation which transforms τ-structures $\mathfrak{A}$ into k-CNF formulas $\mathcal{I}(\mathfrak{A})$, and let $(\mathfrak{A}_n) = (\mathfrak{A}_n)_{n \geq 1}$ be a family of τ-structures. We say that $\mathcal{I}$ *interprets* (Φ_n) *in* $(\mathfrak{A}_n)$ if for every $n \geq 1$, $\Phi_n = \mathcal{I}(\mathfrak{A}_n)$.

▶ **Theorem 17.** *Let $k \geq 1$, let (Φ_n) and (Ψ_n) be two families of k-CNF formulas, let $\mathcal{I}$ be an $L^\ell_{\infty\omega}$-interpretation that maps τ-structures to k-CNF formulas, and let $(\mathfrak{A}_n)$ and $(\mathfrak{B}_n)$ be two families of τ-structures such that:*

- *(Φ_n) consists of satisfiable formulas, and $\mathcal{I}$ interprets (Φ_n) in $(\mathfrak{A}_n)$;*
- *(Ψ_n) consists of unsatisfiable formulas, and $\mathcal{I}$ interprets (Ψ_n) in $(\mathfrak{B}_n)$;*
- *$\mathfrak{A}_n \equiv^{\Omega(n)} \mathfrak{B}_n$.*

(a) *Let* RES-WIDTH(n) *denote the resolution width required to refute the formula Ψ_n. Then* RES-WIDTH$(n) \in \Omega(n)$.

(b) *Let* T-RES-SIZE(n) *denote the size of a tree-like resolution refutation for Ψ_n. Then* T-RES-SIZE(n) *is bounded below by a function in $2^{\Omega(n)}$.*

(c) *Let* RES-SIZE(n) *denote the size of resolution refutation for Ψ_n. If the formulas Ψ_n only contain $\mathcal{O}(n)$ many variables, then* RES-SIZE(n) *is bounded below by a function in $2^{\Omega(n)}$.*

Proof. First, assume that RES-WIDTH$(n) \notin \Omega(n)$. For $r \geq 1$, we choose an EFP-formula ϑ_r, according to Theorem 7, which expresses that a k-CNF formula has a resolution refutation of width r. We can choose ϑ_r in such a way that it only contains $\mathcal{O}(r)$ many variables. By the embedding of EFP into $L^\omega_{\infty\omega}$ we can actually assume that ϑ_r is an $L^\omega_{\infty\omega}$-formula with at most $\mathcal{O}(r)$ many variables. We now translate the formulas ϑ_r back, via $\mathcal{I}$, to τ-structures, that is we obtain $L^\omega_{\infty\omega}$-formulas $\vartheta^{\mathcal{I}}_r$ such that for every τ-structure $\mathfrak{A}$ it holds that $\mathfrak{A} \models \vartheta^{\mathcal{I}}_r$ if, and only if, $\mathcal{I}(\mathfrak{A})$ has a resolution refutation of width r. Since $\mathcal{I}$ is fixed and only contains formulas with $\ell \geq 1$ many variables, the number of variables in the formulas $\vartheta^{\mathcal{I}}_r$ is still bounded by $\mathcal{O}(r)$. For concreteness, assume that $\vartheta^{\mathcal{I}}_r$ contains at most $c \cdot r$ many variables for some constant $c \geq 1$ and all large enough $r \geq 1$. Further, we choose $d > 0$, such that $\mathfrak{A}_n \equiv^{d \cdot n} \mathfrak{B}_n$ for all large enough $n \geq 1$. By our initial assumption we can now choose, for $e := (1/c) \cdot d$, a still larger $n \geq 1$, such that RES-WIDTH$(n) < e \cdot n$. This, however, means that $\mathfrak{A}_n \not\models \vartheta^{\mathcal{I}}_{e \cdot n}$ and $\mathfrak{B}_n \models \vartheta^{\mathcal{I}}_{e \cdot n}$, which implies that $\mathfrak{A}_n \not\equiv^{e \cdot n \cdot c} \mathfrak{B}_n$. This, however, is a contradiction, as $e \cdot n \cdot c = d \cdot n$.

Hence, we know that RES-WIDTH$(n) \in \Omega(n)$. The remaining statements follow from well-known size-width relations for the resolution proof system. In fact, recall from [8], that the size of a smallest tree-like resolution refutation for a k-CNF formula Ψ is bounded below by 2^{w-k} where w denotes the minimal width required to refute Ψ. Since k is a constant, we get a lower bound on T-RES-SIZE(n) as $2^{\Omega(n)}$. Moreover, it is also shown in [8] that the size of a smallest resolution refutation for a k-CNF formula Ψ with m variables is bounded below by $2^{\Omega((w-k)^2/m)}$. Hence, if we additionally have that the number of variables in Ψ_n is at most $\mathcal{O}(n)$, then this gives us a lower bound of $2^{\Omega(n)}$ for RES-SIZE(n). ◀

We remark that the assumptions of the theorem can be generalised in several ways. For instance, one could formulate a similar theorem for the degree of Monomial-PC proofs by replacing logical equivalence in $L^\omega_{\infty\omega}$ by equivalence in counting logic. This would allow us to generalise some of our lower bounds below on resolution width and size to lower bounds on Monomial-PC degree and size. We defer details to a full version of the paper.

Pigeonhole Principle. For $n, m \geq 1$ the pigeonhole principle formulas PHP^m_n express that there is an injective function from a set of size n to a set of size m. The usual definition leads to CNF-formulas of unbounded width. However, one can also formulate the pigeonhole principle using 3-CNF formulas and auxiliary variables, so called extension variables. More precisely, the 3-CNF formula EPHP^m_n contains variables X_{ij}, for $i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$, saying that pigeon i sits in hole j, and auxiliary variables Y_{ij} for every $i \in \{1, \dots, n\}$ and $j \in \{0, \dots, m\}$. The formula EPHP^m_n is built-up using the following formulas:

$$\neg Y_{i0} \wedge \bigwedge_{j=1}^{m} (Y_{ij-1} \vee X_{ij} \vee \neg Y_{ij}) \wedge Y_{im} \qquad \text{for all } i \in \{1, \dots, n\}$$

$$(\neg X_{ij} \vee \neg X_{i'j}) \qquad \text{for all } i, i' \in \{1, \dots, n\}, j \in \{1, \dots, m\}, i \neq i'$$

We consider relational structures $\mathfrak{C}^m_n = ([n] \uplus [m], P = [n], Q = [m], <)$ where $<$ is a linear order on $Q = [m]$. It is straightforward to construct a first-order interpretation $\mathcal{I}$ which interprets the formula EPHP^m_n in the structure $\mathfrak{C}^m_n$ for all $n, m \geq 1$. However, note that we really need the linear order on the set $Q = [m]$ for this. Let $\Phi_n = \mathrm{EPHP}^n_n$ and $\Psi_n = \mathrm{EPHP}^n_{n+1}$, $\mathfrak{A}_n = \mathfrak{C}^n_n$ and $\mathfrak{B}_n = \mathfrak{C}^n_{n+1}$. Then $\mathfrak{A}_n \equiv^n \mathfrak{B}_n$. Hence, all preconditions of Theorem 17 are satisfied (however, the formulas Ψ_n contain more than n^2 many variables).

▶ **Corollary 18.** *The resolution width required to refute the formulas* EPHP^n_{n+1} *is linear in* n*. This implies that the size of treelike resolution refutations is exponential in* n*.*

Three colourability. Given a graph $G = (V, E)$ we can write down a propositional formula $\Theta[G]$ saying that the graph G is 3-colourable. For each vertex $v \in V$ of G, and each colour $i \in \{0, 1, 2\}$ we consider a variable X_v^i indicating that the vertex v is coloured with colour i. Then $\Theta[G]$ consists of the following clauses:

$$(X_v^0 \vee X_v^1 \vee X_v^2) \qquad \text{for every } v \in V$$
$$(\neg X_v^i \vee \neg X_w^i) \qquad \text{for each edge } (v, w) \in E \text{ and colour } i \in \{0, 1, 2\}.$$

Note that $\Theta[G]$ is a 3-CNF formula and the number of variables is linear in $|V|$. Again, it is easy to construct a first-order interpretation $\mathcal{I}$ such that for every graph G it holds that $\mathcal{I}(G) = \Theta[G]$. We make use of the following fact from finite model theory, see Appendix B.

▶ **Theorem 19** ([3, 11, 17, 27]). *For all $n \geq 1$, there are graphs G_n, H_n with $\mathcal{O}(n)$ many vertices, such that $G_n \equiv^{\Omega(n)} H_n$, and such that G_n is three-colourable and H_n is not three-colourable.*

Using this, we can apply Theorem 17: we set $\mathfrak{A}_n = G_n$, $\mathfrak{B}_n = H_n$, $\Phi_n = \Theta[G_n]$, $\Psi_n = \Theta[H_n]$. Also, note that the number of variables in Ψ_n is bounded by $\mathcal{O}(n)$. We get:

▶ **Corollary 20.** *Refuting the formulas $\Theta[H_n]$ for three-colourability requires resolution proofs of exponential size and linear width.*

Graph isomorphism problem. In the introduction we mentioned the graph isomorphism problem. This problem is not known to be decidable in polynomial time, but there is strong evidence that it is not NP-complete. In [26] Toran showed that, with respect to a natural encoding of the graph isomorphism problem as a propositional formula, one cannot obtain an efficient graph isomorphism test based on resolution, not even for graphs with colour class size four. However, one could argue that Toran considered one *specific* encoding of the graph isomorphism problem as a propositional formula only. Maybe one could use a different encoding of the graph isomorphism problem, which may involve simple precomputations, that actually allows efficient resolution refutations? Our results indicate that this is not the case. In fact, Toran's result holds with respect to *every* LFP-definable encoding that uses a linear number of propositional variables only (which is a natural assumption in the context of graphs with constant colour class size), see Appendix A.

5.2 Solving Parity Games via Resolution

To some degree, the complexity-theoretic status of the problem of *computing winning regions in parity games* resembles the situation for graph isomorphism. Most importantly, we do not know whether winning regions in parity games can be computed in polynomial time, but we do not expect that this problem is NP-complete. In fact, for both problems there have been recent breakthroughs providing new algorithms that solve them in quasi-polynomial time [6, 12]. However, there also are notable differences. For example, computing winning regions in parity games is known to be P-hard, but we do not have such strong lower bounds for the graph isomorphism problem. Another remarkable difference between the graph isomorphism problem and the problem of computing winning regions in parity games follows from Theorem 2 and a result due to Dawar and the first author [16], showing that a polynomial-time algorithm for parity games would imply that their winning regions are LFP-definable (despite the fact that LFP is, in general, weaker than polynomial time).

▶ **Theorem 21.** *Parity games can be solved in polynomial time if, and only if, they can be solved using Horn resolution with respect to a first-order definable encoding.*

References

1 M. Anderson and A. Dawar. On symmetric circuits and fixed-point logics. In *STACS 2014*, pages 41–52, 2014.

2 A. Atserias. On sufficient conditions for unsatisfiability of random formulas. *J. ACM*, 51(2):281–311, 2004.

3 A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410:1666–1683, 2009.

4 A. Atserias and V. Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008.

5 A. Atserias and E. N. Maneva. Sherali-adams relaxations and indistinguishability in counting logics. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 367–379. ACM, 2012.

6 L. Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015. URL: `http://arxiv.org/abs/1512.03547`.

7 P. Beame and T. Pitassi. Propositional Proof Complexity: Past, Present, and Future. *Current Trends in TCS: Entering the 21st Century*, pages 42–70, 2001.

8 E. Ben-Sasson and A. Wigderson. Short proofs are narrow – resolution made simple. *J. ACM*, 48(2):149–169, 2001.

9 C. Berkholz and M. Grohe. Limitations of algebraic approaches to graph isomorphism testing. In *Proceedings of ICALP 2015*, pages 155–166, 2015.

10 C. Berkholz and M. Grohe. Linear diophantine equations, group csps, and graph isomorphism. In *Proceedings of SODA 2017*, pages 327–339, 2017.

11 J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

12 C. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. In *STOC 2017*, 2017.

13 M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Groebner Basis Algorithm to find Proofs of Unsatisfiability. In *STOC 1996*, pages 174–183, 1996.

14 S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *J. Symbolic Logic*, 44:36–50, 1979.

15 A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2(1):8–21, 2015.

16 A. Dawar and E. Grädel. The descriptive complexity of parity games. In *Proceedings of CSL 2008*, pages 354–368, 2008.

17 A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *LICS 2009*, pages 113–122, 2009.

18 A. Dawar and P. Wang. Lasserre lower bounds and definability of semidefinite programming. *CoRR*, abs/1602.05409, 2016.

19 M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

20 E. Grädel and S. Hegselmann. Counting in Team Semantics. In *CSL 2016*, 2016.

21 M. Grohe and M. Otto. Pebble games and linear equations. *J. Symb. Log.*, 80(3):797–844, 2015.

22 P. N. Malkin. Sherali-adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 12:73–97, 2014.

23 M. Otto. *Bounded Variable Logics and Counting*. Springer, 1997.

24 W. Pakusa. *Linear Equation Systems and the Search for a Logical Characterisation of Polynomial Time*. PhD thesis, RWTH Aachen University, 2016.

25 N. Segerlind. The Complexity of Propositional Proofs. *Bulletin of Symbolic Logic*, 13(04):417–481, 2007.

26 J. Torán. On the resolution complexity of graph non-isomorphism. In *Theory and Applications of Satisfiability Testing – SAT 2013*, volume 7962 of *LNCS*, pages 52–66, 2013.

27 O. Verbitsky. On the dynamic width of the 3-colorability problem. *CoRR*, abs/1312.5937, 2013.

A Details omitted from Section 5.1

A *graph with colour class size* $k \geq 1$ is a structure $G = (V, E, \preceq)$ where (V, E) is a graph and where $\preceq$ is a linear preorder on V such that every class of $\preceq$-incomparable vertices, that is every *colour class*, is of size at most k. In other words, one can think of the vertices of the graph G to be coloured, but we only allow that at most k many vertices get the same colour. We write $V = V_0 \preceq \cdots \preceq V_{n-1}$ to denote that V is linearly ordered by $\preceq$ into n colour classes V_i in the indicated way. We have that $|V_i| \leq k$ for every $i < n$.

We consider pairs of graphs $G = (V, E, \preceq_V)$ and $H = (W, F, \preceq_W)$ of colour class size $k \geq 1$ with the same number of colour classes, that is

$$V = V_0 \preceq_V V_1 \preceq_V \cdots \preceq_V V_{n-1}$$
$$W = W_0 \preceq_W W_1 \preceq_W \cdots \preceq_W W_{n-1}.$$

We consider the following propositional formula $\Theta(G, H)$ which encodes the graph isomorphism problem for G and H. The formula uses propositional variables $X[v \mapsto w]$ for $v \in V_i$, $w \in W_i$, $i < n$, indicating that the vertex $v \in V_i$ is mapped to the vertex $w \in W_i$ (we only consider colour-preserving mappings). The formula $\Theta(G, H)$ consists of the following set of clauses:

$$\bigvee_{w \in W_i} X[v \mapsto w] \qquad \text{for each } i < n, v \in V_i \qquad (1)$$

$$\bigvee_{v \in V_i} X[v \mapsto w] \qquad \text{for each } i < n, w \in W_i \qquad (2)$$

$$\neg(X[v_1 \mapsto w_1] \wedge X[v_2 \mapsto w_2]) \qquad \text{if } \{(v_1, w_1), (v_2, w_2)\} \text{ is not a local isomorphism.} \qquad (3)$$

This is basically the encoding that Toran used in [26]. It is important to observe that $\Theta(G, H)$ is a k-CNF formula where the number of propositional variables is linear in the number of vertices of the graph (we think of the colour class size k to be fixed). For this encoding, Toran proved, using the well-known construction of Cai, Fürer, and Immerman, that there is a family of pairs of non-isomorphic graphs (G_n, H_n), $n \geq 1$, of degree three and colour class size four for which the resolution refutations of $\Theta(G_n, H_n)$ are of exponential size (in the number of vertices of the graphs G_n, H_n). This is a special case of the following more general result.

▶ **Theorem 22.** *Let $k \geq 1$, and let $\mathcal{I}$ be an* LFP*-interpretation that maps pairs of n-vertex graphs (G, H) of degree three and colour class size four to a k-CNF formula $\mathcal{I}(G, H)$ such that*

- *the number of propositional variables in $\mathcal{I}(G, H)$ is $\mathcal{O}(n)$;*
- *$\mathcal{I}(G, H)$ is satisfiable if, and only if, G and H are isomorphic.*

Then the maximal size of a resolution refutation of the formula $\mathcal{I}(G, H)$ for pairs of non-isomorphic n-vertex graphs (G, H), as above, is bounded below by a function in $2^{\Omega(n)}$.

Proof. Another application of Theorem 17. We first fix for all $n \geq 1$ pairs of non-isomorphic graphs G_n, H_n with $\mathcal{O}(n)$ many vertices, of colour class size four, of degree three, and such

that $G_n \equiv^n H_n$; the existence follows from [11]. It follows that $(G_n, G_n) \equiv^n (G_n, H_n)$. By the embedding of LFP into $L^\omega_{\infty\omega}$ we can view the LFP-interpretation $\mathcal{I}$ as an $L^\ell_{\infty\omega}$-interpretation for some fixed $\ell \geq 1$. Now we set $\mathfrak{A}_n = (G_n, G_n)$, $\mathfrak{B}_n = (G_n, H_n)$, $\Phi_n = \mathcal{I}(\mathfrak{A}_n)$, and $\Psi_n = \mathcal{I}(\mathfrak{B}_n)$. All preconditions of Theorem 17 are satisfied. In particular, the number of variables of the formulas Ψ_n is linear in n by our assumption. This shows that the size of a resolution refutation of Ψ_n is bounded below by a function in $2^{\Omega(n)}$ as claimed. ◀

B Proof of Theorem 19

▶ **Theorem 19** (restated). *For all $n \geq 1$, there are graphs G_n, H_n with $\mathcal{O}(n)$ many vertices, such that $G_n \equiv^{\Omega(n)} H_n$, and such that G_n is three-colourable and H_n is not three-colourable.*

We remark that this result, and our proof sketch below, remain valid if we consider the stronger equivalence with respect to counting logic. Indeed, for this setting the result already appeared in [27] and for the more general setting of constraint satisfaction problems with unbounded width in [18]. For completeness, let us sketch a proof here which shows how to derive this result from [3, 11, 17].

Proof. Again, we start with the Cai-Fürer-Immerman construction [11]. For all $n \geq 1$ we obtain a pair of three-regular, non-isomorphic graphs (G_n, H_n) of colour class size four and with $\mathcal{O}(n)$ many vertices such that $G_n \equiv^n H_n$. We have $(G_n, G_n) \equiv^n (G_n, H_n)$.

Moreover, for pairs of CFI-graphs $(G^*, H^*) \in \{(G_n, H_n), (G_n, G_n) : n \geq 1\}$ it is known that the isomorphism problem reduces, via a first-order interpretation $\mathcal{I}$, to a linear equation system over the field with two elements, see e.g. [17], which can equivalently be formulated as the satisfiability problem of a propositional formula. It can be checked that the resulting propositional formula $\mathcal{I}(G^*, H^*)$ is indeed a 3-CNF formula, since we work with three-regular graphs. Also the number of propositional variables and clauses in $\mathcal{I}(G^*, H^*)$ is linear in the number of vertices of G^* (and H^*), which, in turn, is linear in n. We can now take a standard reduction from 3-SAT to 3-colourability from [19], which is first-order definable. The number of vertices of the resulting graphs is again linear in the number of clauses and variables from the 3-CNF formula. To sum up, there is a first-order interpretation $\mathcal{J}$ which maps pairs of CFI-graphs $(G^*, H^*) \in \{(G_n, H_n), (G_n, G_n)\}$ to graphs $\mathcal{J}(G^*, H^*)$ such that

- the number of vertices of the graphs $\mathcal{J}(G^*, H^*)$ is $\mathcal{O}(n)$;
- $\mathcal{J}(G_n, G_n)$ is three-colourable;
- $\mathcal{J}(G_n, H_n)$ is not three-colourable;
- $\mathcal{J}(G_n, G_n) \equiv^{\Omega(n)} \mathcal{J}(G_n, H_n)$.

This proves our claim. ◀

Validity and Entailment in Modal and Propositional Dependence Logics*

Miika Hannula

Department of Computer Science, University of Auckland, Auckland, New Zealand
m.hannula@auckland.ac.nz

Abstract

The computational properties of modal and propositional dependence logics have been extensively studied over the past few years, starting from a result by Sevenster showing **NEXPTIME**-completeness of the satisfiability problem for modal dependence logic. Thus far, however, the validity and entailment properties of these logics have remained uncharacterised to a great extent. This paper establishes a complete classification of the complexity of validity and entailment in modal and propositional dependence logics. In particular, we address the question of the complexity of validity in modal dependence logic. By showing that it is **NEXPTIME**-complete we refute an earlier conjecture proposing a higher complexity for the problem.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases modal logic, propositional logic, dependence logic, entailment, validity, complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.28

1 Introduction

The notions of dependence and independence are pervasive in various fields of science. Usually these concepts manifest themselves in the presence of *multitudes* (e.g. events or experiments). Dependence logic is a recent logical formalism which, in contrast to others, has exactly these multitudes as its underlying concept [26]. In this article we study dependence logic in the propositional and modal logic context and present a complete classification of the computational complexity of their associated entailment and validity problems.

In first-order logic, the standard formal language behind all mathematics and computer science, dependencies between variables arise strictly from the order of their quantification. Consequently, more subtle forms of dependencies cannot be captured, a phenomenon exemplified by the fact that first-order logic lacks expressions for statements of the form

"for all x there is y, and for all u there is v, such that $R(x, y, u, v)$"

where y and v are to be chosen independently from one another. To overcome this barrier, branching quantifiers of Henkin and independence-friendly logic of Hintikka and Sandu suggested the use of quantifier manipulation [13, 14]. Dependence logic instead extends first-order logic at the atomic level with the introduction of new dependence atoms

$$\mathrm{dep}(x_1, \ldots, x_n) \tag{1}$$

which indicate that the value of x_n depends only on the values of $x_1, \ldots, x_{n-1}$. Dependence atoms are evaluated over *teams*, i.e., sets of assignments which form the basis of *team*

* This work was supported by the Marsden Fund grant UOA1628.

semantics. The concept of team semantics was originally proposed by Hodges in refutation of the view of Hintikka that the logics of imperfect information, such as his independence-friendly logic, escape natural compositional semantics [15]. By the development of dependence logic it soon became evident that team semantics serves also as a connecting link between the aforenementioned logics and the relational database theory. In particular, team semantics enables the extensions of even weaker logics, such as modal and propositional logics, with various sophisticated dependency notions known from the database literature [5, 7, 16, 17]. In this article we consider modal and propositional dependence logics that extend modal and propositional logics with dependence atoms similar to (1), the only exception being that dependence atoms here declare dependencies between propositions [27]. We establish a complete classification of the computational complexity of the associated entailment and validity problems, including a solution to an open problem regarding the complexity of validity in modal dependence logic.

Modal dependence logic was introduced by Väänänen in 2008, and soon after it was shown to enjoy a **NEXPTIME**-complete satisfiability problem [25]. Since then the expressivity, complexity, and axiomatizability properties of modal dependence logic and its variants have been exhaustively studied. Especially the complexity of satisfiability and model checking for modal dependence logic and its variants has been already comprehensively classified [3, 4, 9, 10, 12, 16, 17, 20, 22]. Furthermore, entailment and validity of modal and propositional dependence logics have been axiomatically characterized by Yang and Väänänen in [30, 31, 32] and also by Sano and Virtema in [24]. Against this background it is rather surprising that the related complexity issues have remained almost totally unaddressed. The aim of this article is to address this shortage in research by presenting a complete classification with regards to these questions. A starting point for this endeavour is a recent result by Virtema which showed that the validity problem for propositional dependence logic is **NEXPTIME**-complete [28, 29]. In that paper the complexity of validity for modal dependence logic remained unsettled, although it was conjectured to be harder than that for propositional dependence logic. This conjecture is refuted in this paper as the same exact **NEXPTIME** bound is shown to apply to modal dependence logic as well. Furthermore, we show that this result applies to the extension of propositional dependence logic with quantifiers as well as to the so-called extended modal logic which can express dependencies between arbitrary modal formulae (instead of simple propositions). We also extend our investigations to the entailment problem and show that for both (quantified) propositional and (extended) modal dependence logics this problem is **co-NEXPTIME**$^{\mathbf{NP}}$-complete. Moreover, our investigations show that for modal logic extended with so-called intuitionistic disjunction the associated entailment, validity, and satisfiability problems are all **PSPACE**-complete, which is, in all the three categories the complexity of the standard modal logic. The outcome of these investigations is a complete picture of the validity and entailment properties of modal and propositional dependence logics, summarized in Table 1 in the conclusion section.

The obtained results have interesting consequences. First, combining results from this paper and [25, 29] we observe that similarly to the standard modal logic case the complexity of validity and satisfiability coincide for (extended) modal dependence logic. It is worth pointing out here that satisfiability and validity cannot be seen as each other's duals in the dependence logic context. Dependence logic cannot express negation nor logical implication which renders its associated validity, satisfiability, and entailment problems genuinely different. Secondly, it was previously known that propositional and modal dependence logics deviate on the complexity of their satisfiability problem (**NP**-complete vs. **NEXPTIME**-complete [20, 25], resp.) and that the standard propositional and modal logics differ from one another

on both satisfiability and validity (**NP**-complete/**co-NP**-complete vs. **PSPACE**-complete, resp. [2, 18, 19]). Based on this it is somewhat surprising to find out that modal and propositional dependence logics correspond to one another in terms of the complexity of both their validity and entailment problems.

Organization. This article is organized as follows. In Section 2 we present some notation and background assumptions. In Section 3 we give a short introduction to modal dependence logics, followed by Section 4 which proves **co-NEXPTIME**$^{\mathbf{NP}}$-membership for modal dependence logic entailment. In Section 5 we define (quanfitied) propositional dependence logics, and in the subsequent section we show **co-NEXPTIME**$^{\mathbf{NP}}$-hardness for entailment in this logic. In Section 7 we draw our findings together, followed by Section 8 that is reserved for conclusions. For some of the proofs we refer the reader to Appendix.

2 Preliminaries

We assume that the reader is familiar with the basic concepts of propositional and modal logic, as well as those of computational complexity. Let us note at this point that all the hardness results in the paper are stated under polynomial-time reductions.

Notation. Following the common convention we assume that all our formulae in the team semantics context appear in negation normal form (NNF). We use $p, q, r, \ldots$ to denote propositional variables. For two sequences $\overline{a}$ and $\overline{b}$, we write $\overline{a}\overline{b}$ to denote their concatenation. For a function f and a sequence $(a_1, \ldots, a_n)$ of elements from the domain of f, we denote by $f(a_1, \ldots, a_n)$ the sequence $(f(a_1), \ldots, f(a_n))$. Let ϕ be any formula. Then $\mathsf{Var}(\phi)$ refers to the set of variables appearing in ϕ, and $\mathsf{Fr}(\phi)$ to the set of free variables appearing in ϕ, both defined in the standard way. We sometimes write $\phi(p_1, \ldots, p_n)$ instead of ϕ to emphasize that $\mathsf{Fr}(\phi) = \{p_1, \ldots, p_n\}$. For a subformula ϕ_0 of ϕ and a formula θ, we write $\phi(\theta/\phi_0)$ to denote the formula obtained from ϕ by substituting θ for ϕ_0. We use $\phi^{\perp}$ to denote the NNF formula obtained from $\neg\phi$ by pushing the negation to the atomic level, and sometimes $\phi^{\top}$ to denote ϕ.

3 Modal Dependence Logics

The syntax of *modal logic* (ML) is generated by the following grammar:

$$\phi ::= p \mid \neg p \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid \Box\phi \mid \Diamond\phi. \tag{2}$$

Extensions of modal logic with different dependency notions are made possible via a generalization of the standard Kripke semantics by teams, here defined as sets of worlds. A *Kripke model* over a set of variables V is a tuple $\mathcal{M} = (W, R, \pi)$ where W is a non-empty set of worlds, R is a binary relation over W, and $\pi\colon V \to \mathcal{P}(W)$ is a function that associates each variable with a set of worlds. A *team* T of a Kripke model $\mathcal{M} = (W, R, \pi)$ is a subset of W. For the team semantics of modal operators we define the set of successors of a team T as $R[T] := \{w \in W \mid \exists w' \in T : (w', w) \in R\}$ and the set of legal successor teams of a team T as $R\langle T\rangle := \{T' \subseteq R[T] \mid \forall w \in T\ \exists w' \in T' : (w, w') \in R\}$. The team semantics of modal logic is now defined as follows.

▶ **Definition 1** (Team Semantics of ML). Let ϕ be an ML formula, let $\mathcal{M} = (W, R, \pi)$ be a Kripke model over $V \supseteq \mathsf{Var}(\phi)$, and let $T \subseteq W$. The satisfaction relation $\mathcal{M}, T \models \phi$ is defined as follows:

$$
\begin{aligned}
\mathcal{M}, T \models p &\quad :\Leftrightarrow \quad T \subseteq \pi(p), \\
\mathcal{M}, T \models \neg p &\quad :\Leftrightarrow \quad T \cap \pi(p) = \emptyset, \\
\mathcal{M}, T \models \phi_1 \wedge \phi_2 &\quad :\Leftrightarrow \quad \mathcal{M}, T \models \phi_1 \text{ and } \mathcal{M}, T \models \phi_2, \\
\mathcal{M}, T \models \phi_1 \vee \phi_2 &\quad :\Leftrightarrow \quad \exists T_1, T_2 : T_1 \cup T_2 = T, \mathcal{M}, T_1 \models \phi_1, \\
&\qquad\qquad \text{and } \mathcal{M}, T_2 \models \phi_2, \\
\mathcal{M}, T \models \Diamond \phi &\quad :\Leftrightarrow \quad \exists T' \in R\langle T \rangle : \mathcal{M}, T' \models \phi, \\
\mathcal{M}, T \models \Box \phi &\quad :\Leftrightarrow \quad \mathcal{M}, R[T] \models \phi.
\end{aligned}
$$

We write $\phi \equiv \psi$ to denote that ϕ and ψ are *equivalent*, i.e., for all Kripke models $\mathcal{M}$ and teams T, $\mathcal{M}, T \models \phi$ iff $\mathcal{M}, T \models \phi'$. Let $\Sigma \cup \{\phi\}$ be a set of formulae. We write $\mathcal{M}, T \models \Sigma$ iff $\mathcal{M}, T \models \phi$ for all $\phi \in \Sigma$, and say that Σ *entails* ϕ if for all $\mathcal{M}$ and T, $\mathcal{M}, T \models \Sigma$ implies $\mathcal{M}, T \models \phi$. Let $\mathcal{L}$ be a logic in the team semantics setting. The *entailment problem* for $\mathcal{L}$ is to decide whether Σ entails ϕ (written $\Sigma \models \phi$) for a given finite set of formulae $\Sigma \cup \{\phi\}$ from $\mathcal{L}$. The *validity problem* for $\mathcal{L}$ is to decide whether a given formula $\phi \in \mathcal{L}$ is satisfied by all Kripke models and teams. The *satisfiability problem* for $\mathcal{L}$ is to decide whether a given formula $\phi \in \mathcal{L}$ is satisfied by some Kripke model and a non-empty team[1].

The following flatness property holds for all modal logic formulae. Notice that by $\models_{\mathsf{ML}}$ we refer to the usual satisfaction relation of modal logic.

▶ **Proposition 2** (Flatness [25]). *Let ϕ be a formula in* ML*, let $\mathcal{M} = (W, R, \pi)$ be a Kripke model over $V \supseteq \mathsf{Var}(\phi)$, and let $T \subseteq W$ be a team. Then:*

$$\mathcal{M}, T \models \phi \quad \Leftrightarrow \quad \forall w \in T : \mathcal{M}, w \models_{\mathsf{ML}} \phi.$$

Team semantics gives rise to different extensions of modal logic capable of expressing various dependency notions. In this article we consider dependence atoms that express functional dependence between propositions. To facilitate their associated semantic definitions, we first define for each world w of a Kripke model $\mathcal{M}$ a truth function $w_{\mathcal{M}}$ from ML formulae into $\{0, 1\}$ as follows:

$$w_{\mathcal{M}}(\phi) = \begin{cases} 1 & \text{if } \mathcal{M}, \{w\} \models \phi, \\ 0 & \text{otherwise.} \end{cases}$$

Dependence atom. *Modal dependence logic* (MDL) is obtained by extending ML with *dependence atoms*

$$\mathrm{dep}(\overline{p}, q) \tag{3}$$

where $\overline{p}$ is a sequence of propositional atoms and q is a single propositional atom. Furthermore, we consider *extended dependence atoms* of the form

$$\mathrm{dep}(\overline{\phi}, \psi) \tag{4}$$

where $\overline{\phi}$ is a sequence of ML formulae and ψ is a single ML formula. The extension of ML with atoms of the form (4) is called *extended modal dependence logic* (EMDL). Atoms of

[1] The empty team satisfies all formulae trivially.

the form (3) and (4) indicate that the (truth) value of the formula on the right-hand side is functionally determined by the (truth) values of the formulae listed on the left-hand side. The satisfaction relation for both (3) and (4) is defined accordingly as follows:

$$\mathcal{M}, T \models \mathrm{dep}(\overline{\phi}, \psi) :\Leftrightarrow \forall w, w' \in T : w_{\mathcal{M}}(\overline{\phi}) = w'_{\mathcal{M}}(\overline{\phi}) \text{ implies } w_{\mathcal{M}}(\psi) = w'_{\mathcal{M}}(\psi).$$

For the sake of our proof arguments, we also extend modal logic with predicates. The syntax of *relational modal logic* (RML) is given by the grammar:

$$\phi ::= p \mid {\sim}\phi \mid (\phi \wedge \phi) \mid \Box\phi \mid S(\phi_1, \ldots, \phi_n). \tag{5}$$

The formulae of RML are evaluated over *relational Kripke models* $\mathcal{M} = (W, R, \pi, S_1^{\mathcal{M}}, \ldots, S_n^{\mathcal{M}})$ where each $S_i^{\mathcal{M}}$ is a set of binary sequences of length $\#S_i$, that is, the arity of the relation symbol S_i. We denote by $\mathcal{M}, w \models_{\mathsf{RML}} \phi$ the satisfaction relation obtained by extending the standard Kripke semantics of modal logic as follows:

$$\mathcal{M}, w \models_{\mathsf{RML}} S(\phi_1, \ldots, \phi_n) :\Leftrightarrow (w_{\mathcal{M}}(\phi_1), \ldots, w_{\mathcal{M}}(\phi_n)) \in S^{\mathcal{M}}.$$

Notice also that by $\sim$ we refer to the contradictory negation, e.g., here $\mathcal{M}, w \models_{\mathsf{RML}} {\sim}\phi :\Leftrightarrow \mathcal{M}, w \not\models_{\mathsf{RML}} \phi$. The team semantics for $\sim$ is also defined analogously. Recall that the negation symbol $\neg$ is used only in front of propositions, and $\neg p$ is satisfied by a Kripke model $\mathcal{M}$ and a team T iff in the standard Kripke semantics it is satisfied by all pointed models $\mathcal{M}, w$ where $w \in T$.

In addition to the aforenementioned dependency notions we examine so-called *intuituionistic disjunction* $\otimes$ defined as follows:

$$\mathcal{M}, T \models \phi_1 \otimes \phi_2 \quad :\Leftrightarrow \quad \mathcal{M}, T \models \phi_1 \text{ or } \mathcal{M}, T \models \phi_2. \tag{6}$$

We denote the extension of ML with intuitionistic disjunction $\otimes$ by ML($\otimes$). Notice that the logics MDL and EMDL are expressively equivalent to ML($\otimes$) but exponentially more succinct as the translation of (3) to ML($\otimes$) involves a necessary exponential blow-up [11]. All these logics satisfy the following downward closure property which will be used in the upper bound result.

▶ **Proposition 3** (Downward Closure [3, 27, 31]). *Let ϕ be a formula in* MDL, EMDL, *or* ML($\otimes$), *let $\mathcal{M} = (W, R, \pi)$ be a Kripke model over $V \supseteq \mathsf{Var}(\phi)$, and let $T \subseteq W$ be a team. Then:*

$$T' \subseteq T \text{ and } \mathcal{M}, T \models \phi \quad \Rightarrow \quad \mathcal{M}, T' \models \phi.$$

We can now proceed to the upper bound result which states that the entailment problem for EMDL is decidable in **co-NEXPTIME$^{\mathbf{NP}}$**.

4 Upper Bound for EMDL Entailment

In this section we show that EMDL entailment is in **co-NEXPTIME$^{\mathbf{NP}}$**. The idea is to represent dependence atoms using witnessing functions guessed universally on the left-hand side and existentially on the right-hand side of an entailment problem $\{\phi_1, \ldots, \phi_{n-1}\} \models \phi_n$. This reduces the problem to validity of an RML formula of the form $\phi_1^* \wedge \ldots \wedge \phi_{n-1}^* \to \phi_n^*$ where ϕ_i^* is obtained by replacing in ϕ_i all dependence atoms with relational atoms whose interpretations are bound by the guess. We then extend an Algorithm by Ladner that shows a **PSPACE** upper bound for the validity problem of modal logic [18]. As a novel

algorithmic feature we introduce recursive steps for relational atoms that query to the guessed functions. The **co-NEXPTIME$^{\mathbf{NP}}$** upper bound then follows by a straightforward running time analysis.

We start by showing how to represent dependence atoms using intuitionistic disjunctions defined over witnessing functions. Let $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$ be a sequence of ML formulae and let β be a single ML formula. Then we say that a function $f : \{\top, \bot\}^n \to \{\top, \bot\}$ is a *witness* of $d := \mathrm{dep}(\overline{\alpha}, \beta)$, giving rise to a witnessing ML formula

$$D(f,d) := \bigvee_{a_1,\ldots,a_n \in \{\top,\bot\}} \alpha_1^{a_1} \wedge \ldots \alpha_n^{a_n} \wedge \beta^{f(a_1,\ldots,a_n)}. \tag{7}$$

The equivalence

$$d \equiv \bigotimes_{f\colon \{\top,\bot\}^n \to \{\top,\bot\}} D(f,d) \tag{8}$$

has been noticed in the contexts of MDL and EMDL respectively in [27, 3]. Note that a representation of the sort (8) necessitates that the represented formula, in this case the dependence atom d, has the downward closure property.

To avoid the exponential blow-up involved in both (7) and (8), we instead relate to RML by utilizing the following equivalence:

$$(W, R, \pi) \models_{\mathsf{ML}} D(f,d) \Leftrightarrow (W, R, \pi, S^{\mathcal{M}}) \models_{\mathsf{RML}} S(\overline{\alpha}\beta), \tag{9}$$

where $S^{\mathcal{M}} := \{(a_1, \ldots, a_n, b) \in \{0,1\}^{n+1} \mid f(a_1, \ldots, a_n) = b\}$. Before proceeding to the proof, we need the following simple proposition, based on [29, 31] where the statement has been proven for empty Σ.

▶ **Proposition 4.** *Let Σ be a set of* ML *formulae, and let $\phi_0, \phi_1 \in \mathsf{ML}(\otimes)$. Then $\Sigma \models \phi_0 \otimes \phi_1$ iff $\Sigma \models \phi_0$ or $\Sigma \models \phi_1$.*

Proof. It suffices to show the only-if direction. Assume first that $\mathcal{L} = \mathsf{ML}$, and let $\mathcal{M}_0, T_0$ and $\mathcal{M}_1, T_1$ be counterexamples to $\Sigma \models \phi_0$ and $\Sigma \models \phi_1$, respectively. W.l.o.g. we may assume that $\mathcal{M}_0$ and $\mathcal{M}_1$ are disjoint. Since the truth value of a $\mathsf{ML}(\otimes)$ formula is preserved under taking disjoint unions of Kripke models (see Theorem 6.1.9. in [31], also Proposition 2.13. in [29]) we obtain that $\mathcal{M}, T_0 \models \Sigma \cup \{\sim\phi_0\}$ and $\mathcal{M}, T_1 \models \Sigma \cup \{\sim\phi_1\}$ where $\mathcal{M} = \mathcal{M}_0 \cup \mathcal{M}_1$. By the downward closure property of $\mathsf{ML}(\otimes)$ (Proposition 3), and by the flatness property of ML (Proposition 2), we then obtain that $\mathcal{M}, T \models \Sigma \cup \{\sim\phi_0, \sim\phi_1\}$ where $T = T_0 \cup T_1$. ◀

The proof now proceeds via Lemmata 5 and 6 of which the former constitutes the basis for our alternating exponential-time algorithm. Note that if ϕ is an EMDL formula with k dependence atom subformulae, listed (possibly with repetitions) in $d_1, \ldots d_k$, then we call $\overline{f} = (f_1, \ldots, f_k)$ a *witness sequence* of ϕ if each f_i is a witness of d_i. Furthermore, we denote by $\phi(\overline{f}/\overline{d})$ the ML formula obtained from ϕ by replacing each d_i with $D(f_i, d_i)$.

▶ **Lemma 5.** *Let $\phi_1, \ldots, \phi_n$ be formulae in* EMDL. *Then $\{\phi_1, \ldots, \phi_{n-1}\} \models \phi_n$ iff for all witness sequences $\overline{f}_1, \ldots, \overline{f}_{n-1}$ of $\phi_1, \ldots, \phi_{n-1}$ there is a witness sequence $\overline{f}$ of ϕ_n such that*

$$\{\phi_1(\overline{f}_1/\overline{d}_1), \ldots, \phi_{n-1}(\overline{f}_{n-1}/\overline{d}_{n-1})\} \models \phi_n(\overline{f}_n/\overline{d}_n).$$

Proof. Assume first that ϕ is an arbitrary formula in EMDL, and let $d = \mathrm{dep}(\overline{\alpha}, \beta)$ be a subformula of ϕ. It is straightforward to show that ϕ is equivalent to

$$\bigotimes_{f\colon \{\top,\bot\}^{|\overline{\alpha}|} \to \{\top,\bot\}} \phi(D(f,d)/d).$$

Iterating these substitutions we obtain that $\{\phi_1, \ldots, \phi_{n-1}\} \models \phi_n$ iff

$$\{\bigotimes_{\overline{f}_1} \phi_i(\overline{f}_1/\overline{d}_1), \ldots, \bigotimes_{\overline{f}_{n-1}} \phi_i(\overline{f}_{n-1}/\overline{d}_{n-1})\} \models \bigotimes_{\overline{f}_n} \phi_i(\overline{f}_n/\overline{d}_n), \tag{10}$$

where $\overline{f}_i$ ranges over the witness sequences of ϕ_i. Then (10) holds iff for all $\overline{f}_1, \ldots, \overline{f}_{n-1}$,

$$\{\phi_1(\overline{f}_1/\overline{d}_1), \ldots, \phi_{n-1}(\overline{f}_{n-1}/\overline{d}_{n-1})\} \models \bigotimes_{\overline{f}_n} \phi(\overline{f}_n/\overline{d}_n). \tag{11}$$

Notice that each formula $\phi_i(\overline{f}_i/\overline{d}_i)$ belongs to ML. Hence, by Proposition 4 we conclude that (11) holds iff for all $\overline{f}_1, \ldots, \overline{f}_{n-1}$ there is $\overline{f}_n$ such that

$$\{\phi_1(\overline{f}_1/\overline{d}_1), \ldots, \phi_{n-1}(\overline{f}_{n-1}/\overline{d}_{n-1})\} \models \phi(\overline{f}_n/\overline{d}_n). \tag{12}$$

◀

The next proof step is to reduce an entailment problem of the form (12) to a validity problem of an RML formula over relational Kripke models whose interpretations agree with the guessed functions. For the latter problem we then apply Algorithm 1 whose lines 1-14 and 19-26 constitute an algorithm of Ladner that shows the **PSPACE** upper bound for modal logic satisfiability [18]. Lines 15-18 consider those cases where the subformula is relational. Lemma 6 now shows that, given an oracle A, this extended algorithm yields a $\mathbf{PSPACE}^A$ decision procedure for satisfiability of RML formulae over relational Kripke models whose predicates agree with A. For an oracle set A of words from $\{0,1,\#\}^*$ and k-ary relation symbol R_i, we define $R_i^A := \{(b_1, \ldots, b_k) \in \{0,1\}^k \mid \mathrm{bin}(i)^\frown \# b_1 \ldots b_k \in A\}$. Note that by $a^\frown b$ we denote the concatenation of two strings a and b.

▶ **Lemma 6.** *Given an* RML*-formula* ϕ *over a vocabulary* $\{S_1, \ldots, S_n\}$ *and an oracle set of words* A *from* $\{0,1,\#\}^*$*, Algorithm 1 decides in* $\mathbf{PSPACE}^A$ *whether there is a relational Kripke structure* $\mathcal{M} = (W, R, \pi, S_1^A, \ldots, S_n^A)$ *and a world* $w \in W$ *such that* $\mathcal{M}, w \models_{\mathsf{RML}} \phi$*.*

Proof. We leave it to the reader to show (by a straightforward structural induction) that, given an input $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ where $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D} \subseteq \mathsf{RML}$, Algorithm 1 returns $\mathtt{Sat}(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ true iff there is a relational Kripke model $\mathcal{M} = (W, R, \pi, S_1^A, \ldots, S_n^A)$ such that

- $\mathcal{M}, w \models_{\mathsf{RML}} \phi$ if $\phi \in \mathcal{A}$;
- $\mathcal{M}, w \not\models_{\mathsf{RML}} \phi$ if $\phi \in \mathcal{B}$;
- $\mathcal{M}, w \models_{\mathsf{RML}} \Box\phi$ if $\phi \in \mathcal{C}$; and
- $\mathcal{M}, w \not\models_{\mathsf{RML}} \Box\phi$ if $\phi \in \mathcal{D}$.

Hence, $\mathtt{Sat}(\{\psi\}, \emptyset, \emptyset, \emptyset)$ returns true iff ψ is satisfiable by $\mathcal{M}, w$ with relations $S_i^{\mathcal{M}}$ obtained from the oracle. We note that the selection of subformulae ϕ from $\mathcal{A} \cup \mathcal{B}$ can be made deterministically by defining an ordering for the subformulae. Furthermore, we note that this algorithm algorithm runs in $\mathbf{PSPACE}^A$ as it employs $\mathcal{O}(n)$ recursive steps that each take space $\mathcal{O}(n)$. A detailed space analysis (following that in [18]) can be found in Appendix. ◀

Using Lemmata 5 and 6 we can now show the $\mathbf{co\text{-}NEXPTIME}^{\mathbf{NP}}$ upper bound. In the proof we utilize the following connection between alternating Turing machines and the exponential time hierarchy at the level $\mathbf{co\text{-}NEXPTIME}^{\mathbf{NP}} = \Pi_2^{\mathrm{EXP}}$.

▶ **Theorem 7** ([1, 21])**.** Σ_k^{EXP} *(or* Π_k^{EXP}*) is the class of problems recognizable in exponential time by an alternating Turing machine which starts in an existential (universal) state and alternates at most* $k-1$ *many times.*

Algorithm 1: PSPACE^A algorithm for deciding validity in RML. Notice that queries to S_i^A range over $(b_1, \ldots, b_k) \in \{0,1\}^k$.

Input : $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ where $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D} \subseteq \mathsf{RML}$
Output : $\mathtt{Sat}(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$

if $\mathcal{A} \cup \mathcal{B} \not\subseteq \mathsf{Prop}$ **then**
  choose $\phi \in (\mathcal{A} \cup \mathcal{B}) \setminus \mathsf{Prop}$;
  if $\phi = \sim\psi$ *and* $\phi \in \mathcal{A}$ **then**
    return $\mathtt{Sat}(\mathcal{A} \setminus \{\phi\}, \mathcal{B} \cup \{\psi\}, \mathcal{C}, \mathcal{D})$;
  else if $\phi = \sim\psi$ *and* $\phi \in \mathcal{B}$ **then**
    return $\mathtt{Sat}(\mathcal{A} \cup \{\psi\}, \mathcal{B} \setminus \{\phi\}, \mathcal{C}, \mathcal{D})$;
  else if $\phi = \psi \wedge \theta$ *and* $\phi \in \mathcal{A}$ **then**
    return $\mathtt{Sat}((\mathcal{A} \cup \{\psi, \theta\}) \setminus \{\phi\}, \mathcal{B}, \mathcal{C}, \mathcal{D})$;
  else if $\phi = \psi \wedge \theta$ *and* $\phi \in \mathcal{B}$ **then**
    return $\mathtt{Sat}(\mathcal{A}, (\mathcal{B} \cup \{\psi\}) \setminus \{\phi\}, \mathcal{C}, \mathcal{D}) \vee \mathtt{Sat}(\mathcal{A}, (\mathcal{B} \cup \{\theta\}) \setminus \{\phi\}, \mathcal{C}, \mathcal{D})$;
  else if $\phi = \Box\psi$ *and* $\phi \in \mathcal{A}$ **then**
    return $\mathtt{Sat}(\mathcal{A} \setminus \{\phi\}, \mathcal{B}, \mathcal{C} \cup \{\psi\}, \mathcal{D})$;
  else if $\phi = \Box\psi$ *and* $\phi \in \mathcal{B}$ **then**
    return $\mathtt{Sat}(\mathcal{A}, \mathcal{B} \setminus \{\phi\}, \mathcal{C}, \mathcal{D} \cup \{\psi\})$;
  else if $\phi = S_i(\psi_1, \ldots, \psi_k)$ *and* $\phi \in \mathcal{A}$ **then**
    return $\bigvee_{(b_1,\ldots,b_k) \in S_i^A} \mathtt{Sat}((\mathcal{A} \cup \{\psi_j : b_j = 1\}) \setminus \{\phi\}, \mathcal{B} \cup \{\psi_j : b_j = 0\}, \mathcal{C}, \mathcal{D})$;
  else if $\phi = S_i(\psi_1, \ldots, \psi_k)$ *and* $\phi \in \mathcal{B}$ **then**
    return $\bigvee_{(b_1,\ldots,b_k) \notin S_i^A} \mathtt{Sat}(\mathcal{A} \cup \{\psi_j : b_j = 1\}, (\mathcal{B} \cup \{\psi_j : b_j = 0\}) \setminus \{\phi\}, \mathcal{C}, \mathcal{D})$;
  end
else if $(\mathcal{A} \cup \mathcal{B}) \subseteq \mathsf{Prop}$ **then**
  if $\mathcal{A} \cap \mathcal{B} \neq \emptyset$ **then**
    return false;
  else if $\mathcal{A} \cap \mathcal{B} = \emptyset$ *and* $\mathcal{C} \cap \mathcal{D} \neq \emptyset$ **then**
    return $\bigwedge_{D \in \mathcal{D}} \mathtt{Sat}(\mathcal{C}, \{D\}, \emptyset, \emptyset)$;
  else if $\mathcal{A} \cap \mathcal{B} = \emptyset$ *and* $\mathcal{C} \cap \mathcal{D} = \emptyset$ **then**
    return true;
  end
end

▶ **Theorem 8.** *The entailment problem for* EMDL *is in* **co-NEXPTIME$^{\mathbf{NP}}$**.

Proof. Assuming an input $\phi_1, \ldots, \phi_n$ of EMDL-formulae, we show how to decide in Π_2^{EXP} whether $\{\phi_1, \ldots, \phi_{n-1}\} \models \phi_n$. By Theorem 7 it suffices to construct an alternating exponential-time algorithm that switches once from an universal to an existential state. By Lemma 5, $\{\phi_1, \ldots, \phi_{n-1}\} \models \phi_n$ iff for all $\overline{f}_1, \ldots, \overline{f}_{n-1}$ there is $\overline{f}_n$ such that

$$\{\phi_1(\overline{f}_1/\overline{d}_1), \ldots, \phi_{n-1}(\overline{f}_{n-1}/\overline{d}_{n-1})\} \models \phi(\overline{f}_n/\overline{d}_n). \tag{13}$$

Recall from the proof of Lemma 5 that all the formulae in (13) belong to ML. Hence by the flatness property (Proposition 2) $\models$ is interchangeable with $\models_{\mathsf{ML}}$ in (13). It follows that (13) holds iff

$$\phi := \phi_1(\overline{f}_1/\overline{d}_1) \wedge \ldots \wedge \phi_{n-1}(\overline{f}_{n-1}/\overline{d}_{n-1}) \wedge \sim\phi(\overline{f}_n/\overline{d}_n) \tag{14}$$

is not satisfiable with respect to the standard Kripke semantics of modal logic. By the equivalence in (9) we notice that (14) is not satisfiable with respect to $\models_{\mathsf{ML}}$ iff ϕ^* is not satisfiable over the selected functions with respect to $\models_{\mathsf{RML}}$, where ϕ^* is obtained from ϕ by replacing each $D(f, \overline{\alpha}, \beta)$ of the form (7) with the predicate $f(\overline{\alpha}) = \beta$, and each appearance of $\neg$, $\Diamond$, or $\psi_0 \vee \psi_1$ respectively with $\sim$, $\sim\Box\sim$, or $\sim(\sim\psi_0 \wedge \sim\psi_1)$. The crucial point here is that ϕ^* is only of length $\mathcal{O}(n \log n)$ in the input.

The algorithm now proceeds as follows. The first step is to universally guess functions listed in $\overline{f}_1 \dots \overline{f}_{n-1}$, followed by an existential guess over functions listed in $\overline{f}_n$. The next step is to transform the input to the described RML formula ϕ^*. The last step is to run Algorithm 1 on $\mathtt{Sat}(\phi^*, \emptyset, \emptyset, \emptyset)$ replacing queries to the oracle with investigations on the guessed functions, and return true iff the algorithm returns false. By Lemma 6, Algorithm 1 returns false iff (13) holds over the selected functions. Hence, by Lemma 5 we conclude that the overall algorithm returns true iff $\{\phi_1, \dots, \phi_{n-1}\} \models \phi_n$.

Note that this procedure involves polynomially many guesses, each of at most exponential length. Also, Algorithm 1 runs in exponential time and thus each of its implementations has at most exponentially many oracle queries. Hence, we conclude that the given procedure decides EMDL-entailment in **co-NEXPTIME$^{\mathbf{NP}}$**. ◀

Notice that the decision procedure for $\models \phi$ does not involve any universal guessing. Therefore, we obtain immediately a **NEXPTIME** upper bound for the validity problem of EMDL.

▶ **Corollary 9.** *The validity problem for* EMDL *is in* **NEXPTIME**.

5 Propositional Dependence Logics

We will show that **co-NEXPTIME$^{\mathbf{NP}}$** is also the lower bound for the entailment problem of the propositional fragment of MDL. Before proceeding to the proof we need to formally define this fragment.

The syntax of *propositional logic* (PL) is generated by the following grammar:

$$\phi ::= p \mid \neg p \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \tag{15}$$

The syntax of propositional dependence logic (PDL) is obtained by extending the syntax of PL with dependence atoms of the form (3). Furthermore, the syntax of PL($\mathbin{⫽}$) extends (15) with the grammar rule $\phi ::= \phi \mathbin{⫽} \phi$.

The formulae of these logics are evaluated against propositional teams. Let V be a set of variables. We say that a function $s\colon V \to \{0,1\}$ is a *(propositional) assignment* over V, and a *(propositional) team* X over V is a set of propositional assignments over V. A team X over V induces a Kripke model $\mathcal{M}_X = (T_X, \emptyset, \pi)$ where $T_X = \{w_s \mid s \in X\}$ and $w_s \in \pi(p) \Leftrightarrow s(p) = 1$ for $s \in X$ and $p \in V$. The team semantics for propositional formulae is now defined as follows:

$$X \models \phi :\Leftrightarrow \mathcal{M}_X, T_X \models \phi,$$

where $\mathcal{M}_X, T_X \models \phi$ refers to the team semantics of modal formulae (see Sect. 3). If ϕ^* is a formula obtained from ϕ by replacing all propositional atoms p (except those inside a dependence atom) with predicates $A(p)$, then we can alternatively describe that $X \models \phi$ iff $\mathcal{M} = (\{0,1\}, A := \{1\})$ and X satisfy ϕ^* under the lax team semantics of first-order dependence logics [5].

We will also examine validity and entailment in quantified propositional dependence logic which is a team semantics adaptation and generalization of the dependency quantified

Boolean formula problem [8]. This problem, shown to be **NEXPTIME**-complete in [23], extends the quantified Boolean formula problem, the standard **PSPACE**-complete problem, with the introduction of additional quantification constraints. To this end, we start with the introduction of quantified propositional logic. The syntax *quantified propositional logic* (QPL) is obtained by extending that of PL with universal and existential quantification over propositional variables. Their semantics is given in terms of so-called duplication and supplementation teams. Let p be a propositional variable and s an assignment over V. We denote by $s(a/p)$ the assignment over $V \cup \{p\}$ that agrees with s everywhere, except that it maps p to a. Universal quantification of a propositional variable p is defined in terms of *duplication teams* $X[\{0,1\}/p] := \{s(a/p) \mid s \in X, a \in \{0,1\}\}$ that extend teams X with all possible valuations for p. Existential quantification is defined in terms of *supplementation teams* $X[F/p] := \{s(a/p) \mid s \in X, a \in F(s)\}$ where F is a mapping from X into $\{\{0\},\{1\},\{0,1\}\}$. The supplementation team $X[F/p]$ extends each assignment of X with a non-empty set of values for p. The satisfaction relations $X \models \exists p\phi$ and $X \models \forall p\phi$ are now given as follows:

$$X \models \exists p\phi \quad :\Leftrightarrow \quad \exists F \in {}^{X}\{\{0\},\{1\},\{0,1\}\} : X[F/p] \models \phi,$$
$$X \models \forall p\phi \quad :\Leftrightarrow \quad X[\{0,1\}/p] \models \phi.$$

We denote by QPDL the extension of PDL with quantifiers. Observe that the flatness and downward closure properties of modal formulae (Propositions 2 and 3, resp.) apply now analogously to propositional formulae. Note that by $\models_{\mathsf{PL}}$ we refer to the standard semantics of propositional logic.

▶ **Proposition 10** (Flatness [26]). *Let ϕ be a formula in* QPL*, and let X be a team over $V \supseteq \mathsf{Fr}(\phi)$. Then:*

$$X \models \phi \quad \Leftrightarrow \quad \forall s \in X : s \models_{\mathsf{PL}} \phi.$$

▶ **Proposition 11** (Downward Closure [26]). *Let ϕ be a formula in* QPDL *or* QPL($\emptyset$)*, and let X be a team over a set $V \supseteq \mathsf{Fr}(\phi)$ of propositional variables. Then:*

$$Y \subseteq X \text{ and } X \models \phi \quad \Rightarrow \quad Y \models \phi.$$

We denote the *restriction* of an assignment s to variables in V by $s \upharpoonright V$, and define the restriction of a team X to V, written $X \upharpoonright V$, as $\{s \upharpoonright V \mid s \in X\}$. We conclude this section by noting that, similarly to the first-order case, quantified propositional dependence logic satisfies the following locality property.

▶ **Proposition 12** (Locality [26]). *Let ϕ be a formula in $\mathcal{L}$ where $\mathcal{L} \in \{\mathsf{QPDL}, \mathsf{QPL}(\emptyset)\}$, let X be a team over a set $V \supseteq \mathsf{Fr}(\phi)$, and let $\mathsf{Fr}(\phi) \subseteq V' \subseteq V$. Then:*

$$X \models \phi \quad \Leftrightarrow \quad X \upharpoonright V' \models \phi.$$

6 Lower Bound for PDL Entailment

In this section we prove that the entailment problem for PDL is **co-NEXPTIME**$^{\mathbf{NP}}$-hard. This result is obtained by reducing from a variant of the quantified Boolean formula problem, that is, the standard complete problem for **PSPACE**.

▶ **Definition 13** ([8]). A Σ_k*-alternating dependency quantified Boolean formula* (Σ_k-ADQBF) is a pair $(\phi, \mathcal{C})$ where ϕ is an expression of the form

$$\phi := (\exists f_1^1 \dots \exists f_{j_1}^1)\,(\forall f_1^2 \dots \forall f_{j_2}^2)\,(\exists f_1^3 \dots \exists f_{j_3}^3) \quad \dots \quad (Qf_1^k \dots Qf_{j_k}^k)\,\forall p_1 \dots \forall p_n\,\theta,$$

where $Q \in \{\exists, \forall\}$, $\mathcal{C} = (\bar{c}_1^1, \dots, \bar{c}_{j_k}^k)$ lists sequences of variables from $\{p_1, \dots, p_n\}$, and θ is a quantifier-free propositional formula in which only the quantified variables p_i and function symbols f_j^i with arguments $\bar{c}_j^i$ may appear. Analogously, a Π_k*-alternating dependency quantified Boolean formula* (Π_k-ADQBF) is a pair $(\phi, \mathcal{C})$ where ϕ is an expression of the form

$$\phi := (\forall f_1^1 \dots \forall f_{j_1}^1)\,(\exists f_1^2 \dots \exists f_{j_2}^2)\,(\forall f_1^3 \dots \forall f_{j_3}^3) \quad \dots \quad (Qf_1^k \dots Qf_{j_k}^k)\,\forall p_1 \dots \forall p_n\,\theta,$$

The sequence $\mathcal{C}$ is called the *constraint* of ϕ.

The truth value of a Σ_k-ADQBF or a Π_k-ADQBF instance is determined by interpreting each Qf_j^i where $Q \in \{\exists, \forall\}$ as existential/universal quantification over Skolem functions $f_j^i \colon \{0,1\}^{|\bar{c}_j^i|} \to \{0,1\}$. Let us now denote the associated decision problems by TRUE(Σ_k-ADQBF) and TRUE(Π_k-ADQBF). These problems characterize levels of the exponential hierarchy in the following way.

▶ **Theorem 14** ([8]). *Let $k \geq 1$. For odd k the problem* TRUE(Σ_k-ADQBF) *is* Σ_k^{EXP}*-complete. For even k the problem* TRUE(Π_k-ADQBF) *is* Π_k^{EXP}*-complete.*

Since TRUE(Π_2-ADQBF) is **co-NEXPTIME$^{\mathbf{NP}}$**-complete, we can show the lower bound via an reduction from it. Notice that regarding the validity problem of PDL, we already have the following lower bound.

▶ **Theorem 15** ([29]). *The validity problem for* PDL *is* **NEXPTIME***-complete, and for* MDL *and* EMDL *it is* **NEXPTIME***-hard.*

This result was shown by a reduction from the dependency quantified Boolean formula problem (i.e. TRUE(Σ_1-ADQBF)) to the validity problem of PDL. We use essentially the same technique to reduce from TRUE(Π_2-ADQBF) to the entailment problem of PDL.

▶ **Theorem 16.** *The entailment problem for* PDL *is* **co-NEXPTIME$^{\mathbf{NP}}$***-hard.*

Proof. By Theorem 14 it suffices to show a reduction from TRUE(Π_2-ADQBF). Let $(\phi, \mathcal{C})$ be an instance of Π_2-ADQBF in which case ϕ is of the form

$$\forall f_1 \dots \forall f_m \exists f_{m+1} \dots \exists f_{m+m'} \forall p_1 \dots \forall p_n \theta$$

and $\mathcal{C}$ lists tuples $\bar{c}_i \subseteq \{p_1, \dots, p_n\}$, for $i = 1, \dots, m + m'$. Let q_i be a fresh propositional variable for each Skolem function f_i. We define $\Sigma := \{\mathrm{dep}(\bar{c}_i, q_i) \mid i = 1, \dots, m\}$ and

$$\psi := \theta \vee \bigvee_{i=m+1}^{m+m'} \mathrm{dep}(\bar{c}_i, q_i)\,.$$

Clearly, Σ and ψ can be constructed from $(\phi, \mathcal{C})$ in polynomial time. It suffices to show that $\Sigma \models \psi$ iff ϕ is true.

Assume first that $\Sigma \models \psi$ and let $f_i \colon \{0,1\}^{|\bar{c}_i|} \to \{0,1\}$ be arbitrary for $i = 1, \dots, m$. Construct a team X that consists of all assignments s that map $p_1, \dots, p_n, q_{m+1}, \dots, q_{m+m'}$ into $\{0,1\}$ and $q_1, \dots, q_m$ respectively to $f_1(s(\bar{c}_1)), \dots, f_m(s(\bar{c}_m))$. Since $X \models \Sigma$ we find $Z, Y_1, \dots, Y_{m'} \subseteq X$ such that $Z \cup Y_1 \cup \dots \cup Y_{m'} = X$, $Z \models \theta$, and $Y_i \models \mathrm{dep}(\bar{c}_{m+i}, q_{m+i})$ for

$i = 1, \ldots, m'$. We may assume that each Y_i is a maximal subset satisfying $\mathrm{dep}(\overline{c}_{m+i}, q_{m+i})$, i.e., for all $s \in X \setminus Y_i$, $Y_i \cup \{s\} \not\models \mathrm{dep}(\overline{c}_{m+i}, q_{m+i})$. By downward closure (Proposition 11) we may assume that Z does not intersect any of the subsets $Y_1, \ldots, Y_{m'}$. It follows that there are functions $f_i \colon \{0,1\}^{|\overline{c}_i|} \to \{0,1\}$, for $i = m+1, \ldots, m+m'$, such that

$$Z = \{s\big(f_{m+1}(s(\overline{c}_{m+1}))/q_{m+1}, \ldots, f_{m+m'}(s(\overline{c}_{m+m'}))/q_{m+m'}\big) \mid s \in X\}.$$

Notice that Z is maximal with respect to $p_1, \ldots, p_n$, i.e., $Z \upharpoonright \{p_1, \ldots, p_n\} = {}^{\{p_1,\ldots,p_n\}}\{0,1\}$. Hence, by the flatness property (Proposition 10), and since $Z \models \theta$, it follows that θ holds for all values of $p_1, \ldots, p_n$ and for the values of $q_1, \ldots, q_{m+m'}$ chosen respectively according to $f_1, \ldots, f_{m+m'}$. Therefore, ϕ is true which shows the direction from left to right.

Assume then that ϕ is true, and let X be a team satisfying Σ. Then there are functions $f_i \colon \{0,1\}^{|\overline{c}_i|} \to \{0,1\}$ such that $f(s(\overline{c}_i)) = s(q_i)$ for $s \in X$ and $i = 1, \ldots, m$. Since ϕ is true we find functions $f_i \colon \{0,1\}^{|\overline{c}_i|} \to \{0,1\}$, for $i = m+1, \ldots, m+m'$, such that for all $s \in X$:

$$s[f_{m+1}(s(\overline{c}_{m+1}))/q_{m+1}, \ldots, f_{m+m'}(s(\overline{c}_{m+m'}))/q_{m+m'}] \models \theta. \tag{16}$$

Clearly, $Y_i := \{s \in X \mid s(q_i) \neq f(s(\overline{c}_i))\}$ satisfies $\mathrm{dep}(\overline{c}_i, q_i)$ for $i = m+1, \ldots, m+m'$. Then it follows by (16) and flatness (Proposition 10) that $X \setminus (Y_{m+1} \cup \ldots \cup Y_{m+m'})$ satisfies θ. Therefore, $\Sigma \models \psi$ which concludes the direction from right to left. ◀

7 Entailment in Modal and Propositional Dependence Logics

We may now draw together the main results of Sections 4 and 6. There it was shown that in terms of the entailment problem **co-NEXPTIME$^{\mathbf{NP}}$** is both an upper bound for EMDL and an lower bound for PDL. Therefore, we obtain in Theorem 18 that for all the logics inbetween it is also the exact complexity bound. Theorem 17 indicates that we can count QPDL in this set of logics. The proof of this uses standard reduction methods and is located in Appendix.

▶ **Theorem 17.** *The satisfiability, validity, and entailment problems for* QPDL *are polynomial-time reducible to the satisfiability, validity, and entailment problems for* MDL*, respectively.*

▶ **Theorem 18.** *The entailment problem for* EMDL*,* MDL*,* QPDL*, and* PDL *is* **co-NEXPTIME$^{\mathbf{NP}}$***-complete.*

Proof. The upper bound for EMDL and MDL was shown in Theorem 8, and by Theorem 17 the same upper bound applies to QPDL and PDL. The lower bound for all of the logics comes from Theorem 16. ◀

We also obtain that all the logics inbetween PDL and EMDL are **NEXPTIME**-complete in terms of their validity problem. The proof arises analogously from Corollary 9 and Theorem 15.

▶ **Theorem 19.** *The validity problem for* EMDL*,* MDL*,* QPDL*, and* PDL *is* **NEXPTIME***-complete.*

Recall that this close correspondence between propositional and modal dependence logics only holds with respect to their entailment and validity problems. Satisfiability of propositional dependence logic is only **NP**-complete whereas it is **NEXPTIME**-complete for its modal variant. It is also worth noting that the proof of Theorem 8 gives rise to an alternative proof for the **NEXPTIME** upper bound for MDL (and EMDL) satisfiability, originally proved in [25]. Moreover, the technique can be succesfully applied to ML($\oslash$). The following theorem entails that ML($\oslash$) is no more complex than the ordinary modal logic.

Table 1 Summary of results. The stated complexity classes refer to completeness results.

	satisfiability	validity	entailment
PL	**NP** [2, 19]	**co-NP** [2, 19]	**co-NP** [2, 19]
ML	**PSPACE** [18]	**PSPACE** [18]	**PSPACE** [18]
ML($\otimes$)	**PSPACE** [25]	**PSPACE** [Thm. 20]	**PSPACE** [Thm. 20]
PDL	**NP** [20]	**NEXPTIME** [29]	**co-NEXPTIME$^{\mathbf{NP}}$** [Thm. 18]
QPDL, MDL, EMDL	**NEXPTIME** [23, 25], [Thm. 17]	**NEXPTIME** [Thm. 19]	**co-NEXPTIME$^{\mathbf{NP}}$** [Thm. 18]

▶ **Theorem 20.** *The satisfiability, validity, and entailment problems for* ML($\otimes$) *are* **PSPACE***-complete.*

Proof. The lower bound follows from the Flatness property of ML (Proposition 2) and the **PSPACE**-hardness of satisfiability and validity problems for ML [18]. For the upper bound, it suffices to consider the entailment problem. The other cases are analogous. As in the proof of Lemma 5 (see also Theorem 5.2 in [29]) we reduce ML($\otimes$)-formulae to large disjunctions with the help of appropriate witness functions. For an ML($\otimes$)-formula θ, denote by F_θ the set of all functions that map subformulae $\alpha \otimes \beta$ of θ to either α or β. For each $f \in F_\theta$, we then denote by θ^f the formula obtained from θ by replacing each subformula of the form $\alpha \otimes \beta$ with $f(\alpha \otimes \beta)$. It is straightforward to show that θ is equivalent to $\bigotimes_{f \in F_\theta} \theta^f$. Let now $\phi_1, \ldots, \phi_n$ be a sequence of ML($\otimes$) formulae. Analogously to the proof of Lemma 5 we can show using Proposition 4 that $\{\phi_1, \ldots, \phi_{n-1}\} \models \phi_n$ iff for all $f_1 \in F_{\phi_1}, \ldots, f_{n-1} \in F_{\phi_{n-1}}$ there is $f_n \in F_{\phi_n}$ such that $\{\phi_1^{f_1}, \ldots, \phi_{n-1}^{f_{n-1}}\} \models \phi_n^{f_n}$. Notice that the number of intuitionistic disjunctions appearing in $\phi_1, \ldots, \phi_n$ is polynomial, and hence any single sequence of functions $f_1 \in F_{\theta_1}, \ldots, f_n \in F_{\theta_n}$ can be stored using only a polynomial amount of space. It follows that the decision procedure presented in the proof of Theorem 8 can be now implemented in polynomial space. We immediately obtain the **PSPACE** upper bound for validity. For satisfiability, notice that $\bigotimes_{f \in F_\theta} \theta^f$ is satisfiable iff θ^f is satisfiable for some $f \in F_\theta$. Checking the right-hand side can be done as described above. This concludes the proof. ◀

Combining the proofs of Theorem 8 and Theorem 20 we also notice that satifiability, validity, and entailment can be decided in **PSPACE** for EMDL-formulae whose dependence atoms are of *logarithmic length*.

8 Conclusion

We have examined the validity and entailment problem for modal and propositional dependence logics (see Table 1). We showed that the entailment problem for (extended) modal and (quantified) propositional dependence logic is **co-NEXPTIME$^{\mathbf{NP}}$**-complete, and that the corresponding validity problems are **NEXPTIME**-complete. We also showed that modal logic extended with intuitionistic disjunction is **PSPACE**-complete with respect to its satisfiability, validity, and entailment problems, therefore being not more complex than the standard modal logic.

References

1 Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.

2 Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC'71, pages 151–158, New York, NY, USA, 1971. ACM.

3 Johannes Ebbing, Lauri Hella, Arne Meier, Julian-Steffen Müller, Jonni Virtema, and Heribert Vollmer. Extended modal dependence logic. In *Logic, Language, Information, and Computation – 20th International Workshop, WoLLIC 2013, Darmstadt, Germany, August 20-23, 2013. Proceedings*, pages 126–137, 2013. doi:10.1007/978-3-642-39992-3_13.

4 Johannes Ebbing and Peter Lohmann. Complexity of model checking for modal dependence logic. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *Lecture Notes in Computer Science*, pages 226–237. Springer, 2012.

5 Pietro Galliani. Inclusion and exclusion dependencies in team semantics: On some logics of imperfect information. *Annals of Pure and Applied Logic*, 163(1):68–84, 2012.

6 Pietro Galliani, Miika Hannula, and Juha Kontinen. Hierarchies in independence logic. In *Computer Science Logic 2013 (CSL 2013)*, volume 23, pages 263–280, 2013.

7 Erich Grädel and Jouko Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013. doi:10.1007/s11225-013-9479-2.

8 Miika Hannula, Juha Kontinen, Martin Lück, and Jonni Virtema. On quantified propositional logics and the exponential time hierarchy. In *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016.*, pages 198–212, 2016.

9 Miika Hannula, Juha Kontinen, Jonni Virtema, and Heribert Vollmer. Complexity of propositional independence and inclusion logic. In *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, pages 269–280, 2015. doi:10.1007/978-3-662-48057-1_21.

10 Lauri Hella, Antti Kuusisto, Arne Meier, and Heribert Vollmer. Modal inclusion logic: Being lax is simpler than being strict. In *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, pages 281–292, 2015. doi:10.1007/978-3-662-48057-1_22.

11 Lauri Hella, Kerkko Luosto, Katsuhiko Sano, and Jonni Virtema. The expressive power of modal dependence logic. In *Advances in Modal Logic 10, invited and contributed papers from the 10th Conference on "Advances in Modal Logic," held in Groningen, The Netherlands, August 5-8, 2014*, pages 294–312, 2014.

12 Lauri Hella and Johanna Stumpf. The expressive power of modal logic with inclusion atoms. In *Proceedings Sixth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2015, Genoa, Italy, 21-22nd September 2015.*, pages 129–143, 2015. doi:10.4204/EPTCS.193.10.

13 Leon Henkin. Some Remarks on Infinitely Long Formulas. In *Infinitistic Methods. Proc. Symposium on Foundations of Mathematics*, pages 167–183. Pergamon Press, 1961.

14 Jaakko Hintikka and Gabriel Sandu. Informational independence as a semantic phenomenon. In J. E. Fenstad, I. T. Frolov, and R. Hilpinen, editors, *Logic, methodology and philosophy of science*, pages 571–589. Elsevier, 1989.

15 Wilfrid Hodges. Compositional Semantics for a Language of Imperfect Information. *Journal of the Interest Group in Pure and Applied Logics*, 5 (4):539–563, 1997.

16 Juha Kontinen, Julian-Steffen Müller, Henning Schnoor, and Heribert Vollmer. Modal independence logic. In *Advances in Modal Logic 10, invited and contributed papers from the 10th Conference on "Advances in Modal Logic," held in Groningen, The Netherlands, August 5-8, 2014*, pages 353–372, 2014.

17 Andreas Krebs, Arne Meier, and Jonni Virtema. A team based variant of CTL. In *22nd International Symposium on Temporal Representation and Reasoning, TIME 2015, Kassel, Germany, September 23-25, 2015*, pages 140–149, 2015.

18 Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977. doi:10.1137/0206033.

19 Leonid A. Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

20 Peter Lohmann and Heribert Vollmer. Complexity results for modal dependence logic. *Studia Logica*, 101(2):343–366, 2013.

21 Martin Lück. Complete problems of propositional logic for the exponential hierarchy. *CoRR*, abs/1602.03050, 2016. URL: http://arxiv.org/abs/1602.03050.

22 Julian-Steffen Müller and Heribert Vollmer. Model checking for modal dependence logic: An approach through post's lattice. In *Logic, Language, Information, and Computation – 20th International Workshop, WoLLIC 2013, Darmstadt, Germany, August 20-23, 2013. Proceedings*, pages 238–250, 2013.

23 Gary L. Peterson, John H. Reif, and Salman Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7-8):957–992, April 2001.

24 Katsuhiko Sano and Jonni Virtema. Axiomatizing propositional dependence logics. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, pages 292–307, 2015.

25 Merlijn Sevenster. Model-theoretic and computational properties of modal dependence logic. *Journal of Logic and Computation*, 19(6):1157–1173, 2009.

26 Jouko Väänänen. *Dependence Logic*. Cambridge University Press, 2007.

27 Jouko Väänänen. Modal Dependence Logic. In Krzysztof R. Apt and Robert van Rooij, editors, *New Perspectives on Games and Interaction*. Amsterdam University Press, Amsterdam, 2008.

28 Jonni Virtema. Complexity of validity for propositional dependence logics. In *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014.*, pages 18–31, 2014.

29 Jonni Virtema. Complexity of validity for propositional dependence logics. *Inf. Comput.*, 253:224–236, 2017. doi:10.1016/j.ic.2016.07.008.

30 F. Yang. Modal Dependence Logics: Axiomatizations and Model-theoretic Properties. *ArXiv e-prints*, October 2016. arXiv:1610.02710.

31 Fan Yang. On extensions and variants of dependence logic. PhD thesis, University of Helsinki, 2014.

32 Fan Yang and Jouko Väänänen. Propositional logics of dependence. *Annals of Pure and Applied Logic*, 167(7):557–589, 2016. doi:10.1016/j.apal.2016.03.003.

A Appendix

A.1 From Quantified Propositional to Modal Logics

In this section we show how to generate simple polynomial-time reductions from quantified propositional dependence logics to modal dependence logics with respect to their entailment and validity problem. First we present Lemma 21 which is a direct consequence of [6, Lemma 14] that presents prenex normal form translations in the first-order dependence logic setting over structures with universe size at least 2. The result follows by the obvious first-order interpretation of quantified propositional formulae: satisfaction of a quantified propositional formula ϕ by a binary team X can be replaced with satisfaction of ϕ^* by

$\mathcal{M} := (\{0,1\}, t^{\mathcal{M}} := 1, f^{\mathcal{M}} := 0)$ and X, where ϕ^* is a formula obtained from ϕ by replacing atomic propositional formulae p and $\neg p$ respectively with $p = t$ and $p = f$.

▶ **Lemma 21** ([6]). *Any formula ϕ in L, where $\mathsf{L} \in \{\mathsf{QPDL}, \mathsf{QPLInc}, \mathsf{QPLInd}\}$, is logically equivalent to a polynomial size formula $Q_1 p_1 \dots Q_n p_n \psi$ in L where ψ is quantifier-free and $Q_i \in \{\exists, \forall\}$ for $i = 1, \dots, n$.*

Next we show how to describe in modal terms a quantifier block $Q_1 p_1 \dots Q_n p_n$. Using the standard method in modal logic we construct a formula $\mathsf{tree}(V, p, n)$ that enforces the complete binary assignment tree over $p_1, \dots, p_n$ for a team over V [18]. The formulation of $\mathsf{tree}(V, p, n)$ follows the presented in [8]. We define $\mathsf{store}_n(p) := (p \wedge \Box^n p) \vee (\neg p \wedge \Box^n \neg p)$, where $\Box^n$ is a shorthand for $\overbrace{\Box \cdots \Box}^{n \text{ many}}$, to impose the existing values for p to successors in the tree. We also define $\mathsf{branch}_n(p) := \Diamond p \wedge \Diamond \neg p \wedge \Box \mathsf{store}_n(p)$ to indicate that there are ≥ 2 successor states which disagree on the variable p and that all successor states preserve their values up to branches of length n. Then we let

$$\mathsf{tree}(V, p, n) := \bigwedge_{q \in V} \bigwedge_{i=1}^{n} \mathsf{store}_n(q) \wedge \bigwedge_{i=0}^{n-1} \Box^i \mathsf{branch}_{n-(i+1)}(p_{i+1}).$$

Notice that $\mathsf{tree}(V, p, n)$ is an ML-formula and hence has the flatness property by Proposition 2.

▶ **Theorem 17.** *The satisfiability, validity, and entailment problems for QPDL are polynomial-time reducible to the satisfiability, validity, and entailment problems for MDL, respectively.*

Proof. Consider first the entailment problem, and assume that $\Sigma \cup \{\phi\}$ is a finite set of formulae in either QPDL, QPLInd, or $\mathsf{QPLInc}\}$. By Lemma 21 each formula in $\theta \in \Sigma \cup \{\phi\}$ can be transformed in polynomial time to the form $\theta_0 = Q_1 p_1 \dots Q_n p_n \psi$ where ψ is quantifier-free. Moreover, by locality principle (Proposition 12) we may assume that the variable sequences $p_1, \dots, p_n$ corresponding to these quantifier blocks are initial segments of a shared infinite list $p_1, p_2, p_3, \dots$ of variables. Assume m is the maximal length of the quantifier blocks that appear in any of the translations, and let V be the set of variables that appear free in some of them. W.l.o.g. we may assume that $\{p_1, \dots, p_m\}$ and V are disjoint. We let θ_1 be obtained from θ_0 by replacing quantifiers $\exists$ and $\forall$ respectively with $\Diamond$ and $\Box$. It follows that $\Sigma \models \phi$ iff $\{\theta_1 \mid \theta \in \Sigma\} \cup \{\mathsf{tree}(V, p, n)\} \models \phi_1$.[2] For the validity problem, we observe that $\models \phi$ iff $\models \mathsf{tree}(V, p, n) \vee (\mathsf{tree}(V, p, n) \wedge \phi_1)$. Furthermore, for the satisfiability problem we have that ϕ is satisfiable iff $\mathsf{tree}(V, p, n) \wedge \phi_1$ is. Since the reductions are clearly polynomial, this concludes the proof. ◀

B Space Analysis for Algorithm 1

Following [18] we show that Algorithm 1 requires only $\mathcal{O}(n^2)$ space on an input of the form $\mathtt{Sat}(\{\phi\}, \emptyset, \emptyset, \emptyset)$: it takes $\mathcal{O}(n)$ recursive steps, each taking space $\mathcal{O}(n)$.

Size of esch recursive step. At each recursive step $\mathtt{Sat}(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ is stored onto the work tape by listing all subformulae in $\mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{D}$ in such a way that each subformula ψ has

[2] Notice that the direction from left to right does not hold under the so-called strict team semantics where $\exists$ and $\Diamond$ range over individuals. These two logics are not downwards closed and the modal translation does not prevent the complete binary tree of having two distinct roots that agree on the variables in V.

its major connective (or relation/proposition symbol for atomic formulae) replaced with a special marker which also points to the position of the subset where ψ is located. In addition we store at each disjunctive/conjunctive recursive step the subformula or binary number that points to the disjunct/conjunct under consideration. Each recursive step takes now space $\mathcal{O}(n)$.

Number of recursive steps. Given a set of formulae $\mathcal{A}$, we write $|\mathcal{A}|$ for $\Sigma_{\phi\in\mathcal{A}}|\phi|$ where $|\phi|$ is the length of ϕ. We show by induction on $n = |\mathcal{A}\cup\mathcal{B}\cup\mathcal{C}\cup\mathcal{D}|$ that $\mathtt{Sat}(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D})$ has $2n+1$ levels of recursion. Assume that the claim holds for all natural numbers less than n, and assume that $\mathtt{Sat}(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D})$ calls $\mathtt{Sat}(\mathcal{A}',\mathcal{B}',\mathcal{C}',\mathcal{D}')$. Then $|\mathcal{A}'\cup\mathcal{B}'\cup\mathcal{C}'\cup\mathcal{D}'| < n$ except for the case where $\mathcal{A}\cap\mathcal{B}$ is empty and $\mathcal{C}\cap\mathcal{D}$ is not. In that case it takes at most one extra recursive step to reduce to a length $< n$. Hence, by the induction assumption the claim follows. We conclude that the space requirement for Algorithm 1 on $\mathtt{Sat}(\{\phi\},\emptyset,\emptyset,\emptyset)$ is $\mathcal{O}(n^2)$.

CALF: Categorical Automata Learning Framework*

Gerco van Heerdt[1], Matteo Sammartino[2], and Alexandra Silva[3]

1 **University College London, London, UK**
`gerco.heerdt@ucl.ac.uk`
2 **University College London, London, UK**
`m.sammartino@ucl.ac.uk`
3 **University College London, London, UK**
`alexandra.silva@ucl.ac.uk`

Abstract

Automata learning is a technique that has successfully been applied in verification, with the automaton type varying depending on the application domain. Adaptations of automata learning algorithms for increasingly complex types of automata have to be developed from scratch because there was no abstract theory offering guidelines. This makes it hard to devise such algorithms, and it obscures their correctness proofs. We introduce a simple category-theoretic formalism that provides an appropriately abstract foundation for studying automata learning. Furthermore, our framework establishes formal relations between algorithms for learning, testing, and minimization. We illustrate its generality with two examples: deterministic and weighted automata.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases automata learning, category theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.29

1 Introduction

Automata learning enables the use of model-based verification methods on black-box systems. Learning algorithms have been successfully applied to find bugs in implementations of network protocols [15], to provide defense mechanisms against botnets [13], to rejuvenate legacy software [27], and to learn an automaton describing the errors in a program up to a user-defined abstraction [12]. Many learning algorithms were originally designed to learn a deterministic automaton and later, due to the demands of the verification task at hand, extended to other types of automata. For instance, the popular $L^\star$ algorithm, devised by Angluin [2], has been adapted for various other types of automata accepting regular languages [9, 3], as well as more expressive automata, including Büchi-style automata [23, 4], register automata [10, 11], and nominal automata [24]. As the complexity of these automata increases, the learning algorithms proposed for them tend to become more obscure. More worryingly, the correctness proofs become involved and harder to verify.

This paper aims to introduce an abstract framework for studying automata learning algorithms in which specific instances, correctness proofs, and optimizations can be derived without much effort. The framework can also be used to study algorithms such as minimization and testing, showing the close connections between the seemingly different problems. These connections also enable the transfer of extensions and optimizations among algorithms.

* This work was partially supported by the ERC Starting Grant ProFoundNet (grant code 679127).

First steps towards a categorical understanding of active learning appeared in [19]. The abstract definitions provided were based on very concrete data structures used in $\mathtt{L}^\star$, which then restricted potential generalization to study other learning algorithms and capture other data structures used by more efficient variations of the $\mathtt{L}^\star$ algorithm. Moreover, there was no correctness proof, and optimizations and connections to minimization or testing were not explored before in the abstract setting. In the present paper, we develop a rigorous *categorical automata learning framework* (CALF) that overcomes these limitations and can be more widely applied. We start by giving a general overview of the paper and its results.

2 Overview

In this section we provide an overview of CALF. We first establish some global notation, after which we introduce the basic ingredients of active automata learning. Finally, we sketch the structure of CALF and highlight our main results. The reader is assumed to be familiar with elementary category theory and the theory of regular languages.

Notation. The set of words over an alphabet A is written A^*; $A^{\leq n}$ contains all words of length up to $n \in \mathbb{N}$. We denote the empty word by ε and concatenation by $\cdot$ or juxtaposition. We extend concatenation to sets of words $U, V \subseteq A^*$ using $U \cdot V = \{uv \mid u \in U, v \in V\}$. Languages $\mathcal{L} \subseteq A^*$ will often be represented as their characteristic functions $\mathcal{L} \colon A^* \to 2$. Given sets X and Y, we write Y^X for the set of functions $X \to Y$. The set $1 = \{*\}$, where $*$ is an arbitrary symbol, is sometimes used to represent elements $x \in X$ of a set X as functions $x \colon 1 \to X$. We write $|X|$ for the size of a set X.

Active Automata Learning. *Active* automata learning is a widely used technique to learn an automaton model of a system from observations. It is based on direct interaction of the learner with an *oracle* that can answer different types of queries about the system. This is in contrast with *passive* learning, where a fixed set of positive and negative examples is provided and no interaction with the system is possible.

Most active learning algorithms assume the ability to perform *membership queries*, where the oracle is asked whether a certain word belongs to the target language $\mathcal{L}$ over a finite alphabet A. The algorithms we focus on use this ability to approximate the construction of the minimal DFA for $\mathcal{L}$. This construction, due to Nerode [26], can be described as follows. The states of the minimal DFA accepting $\mathcal{L}$ are given by the image of the function $t_{\mathcal{L}} \colon A^* \to 2^{A^*}$ defined by $t_{\mathcal{L}}(u)(v) = \mathcal{L}(uv)$, assigning to each word the residual language after reading that word. More precisely, the minimal DFA for $\mathcal{L}$ is given by the four components

$$M = \{t_{\mathcal{L}}(u) \mid u \in A^*\} \qquad m_0 = t_{\mathcal{L}}(\varepsilon)$$
$$F_M = \{t_{\mathcal{L}}(u) \mid u \in A^*, \mathcal{L}(u) = 1\} \qquad \delta_M(t_{\mathcal{L}}(u), a) = t_{\mathcal{L}}(ua),$$

where M is a finite set of states, $\delta_M \colon M \times A \to M$ is the transition function, $F_M \subseteq M$ is the set of accepting states, and $m_0 \in M$ is the initial state. Note that the transition function is well-defined because of the definition of $t_{\mathcal{L}}$ and M is finite because the language is regular.

Though we know M is finite, computing it naively as the image of $t_{\mathcal{L}}$ does not terminate, since the domain of $t_{\mathcal{L}}$ is the infinite set A^*. Learning algorithms approximate M by instead constructing two finite sets $S, E \subseteq A^*$, which induce the function $\mathsf{row} \colon S \cup S \cdot A \to 2^E$ defined as $\mathsf{row}(u)(v) = \mathcal{L}(uv)$. The name "row" reflects the usual representation of this function as a table with rows indexed by $S \cup S \cdot A$ and columns indexed by E. This table is called an

observation table, and it induces a *hypothesis* DFA given by

$$H = \{\mathsf{row}(s) \mid s \in S\} \qquad h_0 = \mathsf{row}(\varepsilon)$$
$$F_H = \{\mathsf{row}(s) \mid s \in S, \mathcal{L}(s) = 1\} \qquad \delta_H(\mathsf{row}(s), a) = \mathsf{row}(sa).$$

It is often required that ε is an element of both S and E, so that the initial and accepting states are well-defined. For the transition function to be well-defined, the observation table is required to satisfy two properties. These properties were introduced by Gold [16]; we use the names given by Angluin [2].

- **Closedness** states that each transition actually leads to a state of the hypothesis. That is, the table is closed if for all $t \in S \cdot A$ there is an $s \in S$ such that $\mathsf{row}(s) = \mathsf{row}(t)$.
- **Consistency** states that there is no ambiguity in determining the transitions. That is, the table is consistent if for all $s_1, s_2 \in S$ such that $\mathsf{row}(s_1) = \mathsf{row}(s_2)$ we have $\mathsf{row}(s_1 a) = \mathsf{row}(s_2 a)$ for any $a \in A$.

Gold [16] also explained how to update S and E to satisfy these properties. If closedness does not hold, then there exists $sa \in S \cdot A$ such that $\mathsf{row}(s') \neq \mathsf{row}(sa)$ for all $s' \in S$. This is resolved by adding sa to S. If consistency does not hold, then there are $s_1, s_2 \in S$, $a \in A$, and $e \in E$ such that $\mathsf{row}(s_1) = \mathsf{row}(s_2)$, but $\mathsf{row}(s_1 a)(e) \neq \mathsf{row}(s_2 a)(e)$. We add ae to E to distinguish $\mathsf{row}(s_1)$ from $\mathsf{row}(s_2)$. In both cases, the size of the hypothesis increases. Since the hypothesis cannot be larger than M, the table will eventually be closed and consistent.

The hypothesis is an approximation. Hence, the language it accepts may not be the target language. Gold [16] showed, based on earlier algorithms [17, 5], that for any two chains of languages $S_1 \subseteq S_2 \subseteq \cdots$ and $E_1 \subseteq E_2 \subseteq \cdots$, both in the limit equaling A^*, the following holds. There exists an $i \in \mathbb{N}$ such that all hypothesis automata derived from S_j, E_j $(j \geq i)$ are isomorphic to M. These hypotheses are built after enforcing closedness and consistency for the sets S_j and E_j. Unfortunately, the convergence point i is not known to the learner.

Arbib and Zeiger. To obtain a terminating algorithm, an additional assumption is necessary [25]. Arbib and Zeiger [5] explained the algorithm of Ho [17] and rephrased it for DFAs. The algorithm assumes an upper bound n on the size of M. One then simply takes $S = E = A^{\leq n-1}$ and obtains a DFA isomorphic to M as the hypothesis.

ID. Angluin [1] showed that, for algorithms with just this extra assumption, the number of membership queries has a lower bound that is exponential in the size of M. She then introduced an algorithm called `ID` that makes a stronger assumption. It assumes a finite set S of words such that each state of M (that does not accept the empty language) is reached by reading a word in S. Initializing $E = \{\varepsilon\}$, the algorithm obtains M up to isomorphism after making the table consistent (for a slightly stronger definition of consistency related to the sink state being kept implicit). This makes `ID` polynomial in the sizes of M, S, and A.

L*. Subsequently, Angluin introduced the algorithm `L*` [2], which makes yet another assumption: it assumes an oracle that can test any DFA for equivalence with the target language. In the case of a discrepancy, the oracle will provide a *counterexample*, which is a word in the symmetric difference of the two languages. `L*` adds the given counterexample and its prefixes to S. One can show that this means that the next hypothesis will classify the counterexample correctly and that therefore the size of the hypothesis must have increased. The algorithm is polynomial in the sizes of M and A and in the length of the longest counterexample.

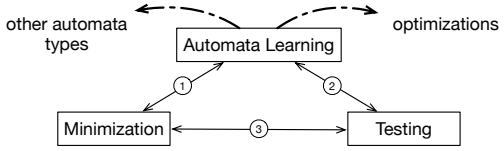


Figure 1 Overview of CALF.

CALF. The framework we introduce in this paper aims to provide a formal and general account of the concepts introduced above, based on category theory. Although the focus is on automata learning, the general perspective brought by CALF allows us to unveil deep connections between learning and algorithms for minimization and testing.

The structure of CALF is depicted in Figure 1. After introducing the basic elements of the framework in Section 3, we use the framework in Section 4 to give a new categorical correctness proof, encompassing all the algorithms mentioned above. Then we explore connections with minimization and testing, following the edges of the triangle in Figure 1:

① Section 5 shows how our correctness proof also applies to minimization algorithms. In particular, we connect consistency fixing to the steps in the classical partition refinement algorithm that merge observationally equivalent states. Dually, reachability analysis, which is also needed to get the minimal automaton, is related to fixing closedness defects.

② Section 6 extends the correctness proof to an abstract result about testing and we show how it applies to algorithms such as the W-method [14], which enable testing against a black-box system.

③ From the abstract framework and the observations in ① and ②, we also see how certain basic sets used in testing algorithms like the W-method can be obtained using generalized minimization algorithms.

This triangular structure is on its own a contribution of the paper, tying together three important techniques that play a role in automata learning and verification. Another strong point of CALF is the ability to seamlessly accommodate variants of learning algorithms:

- algorithms for *other types of automata*, by varying the category on which automata are defined. This includes weighted automata, which are defined in the category of vector spaces. We present details of this example in Appendix B.
- algorithms with *optimizations*, by varying the data structure. Section 7 discusses *classification trees* [20], which optimize the observation tables of classical learning algorithms.

CALF also provides guidelines on how to combine these variants, and on how to obtain corresponding minimization/testing algorithms via the connections described above. This can lead to more efficient algorithms, and, because of our abstract approach, correctness proofs can be reused.

Some proofs have been omitted for space reasons. They are found in Appendix A.

3 Abstract Learning

In this section, we develop CALF by starting with categorical definitions that capture the basic data structures and definitions used in learning.

3.1 Observation Tables as Approximated Algebras

In the automata learning algorithms described in Section 2, observation tables are used to approximate the minimal automaton M for the target language $\mathcal{L}$. We start by introducing a notion of approximation of an object in a category, called *wrapper*. We work in a fixed category $\mathbf{C}$ unless otherwise indicated.

▶ **Definition 1** (Wrapper). A *wrapper for an object* T is a pair of morphisms $w = (S \xrightarrow{\sigma} T, T \xrightarrow{\pi} P)$; T is called the *target* of w.

▶ **Example 2.** The algorithms explained in Section 2 work by producing subsequent approximations of M. In each approximation, we have two finite sets of words $S, E \subseteq A^*$, yielding a wrapper $w = (S \xrightarrow{\sigma} M, M \xrightarrow{\pi} 2^E)$ in **Set**. Here σ maps $s \in S$ to the state reached by M after reading s while π assigns to each state of M the language accepted by that state, but restricted to the words in E. Although M is unknown, the composition of π and σ is given by $(\pi \circ \sigma)(s)(e) = \mathcal{L}(se)$, which can be determined using membership queries. This composition is precisely the component $S \to 2^E$ of the observation table $\mathsf{row}\colon S \cup S \cdot A \to 2^E$.

We claim that the other component $S \cdot A \to 2^E$ of row is an approximated version of the transition function $\delta_M\colon M \times A \to M$, and that it can be obtained via the wrapper w. More generally, we can abstract away from the structure of automata and approximate algebras for a functor F, i.e., pairs $(T, f\colon FT \to T)$. Notice that (M, δ_M) is an algebra for $(-) \times A$.

▶ **Definition 3** (Approximation of Algebras Along a Wrapper). Given an F-algebra (T, f) and a wrapper $w = (S \xrightarrow{\sigma} T, T \xrightarrow{\pi} P)$, the *approximation of f along w* is $\xi^w_f = FS \xrightarrow{F\sigma} FT \xrightarrow{f} T \xrightarrow{\pi} P$. We write ξ^w for $\xi^w_{\mathsf{id}} = \pi \circ \sigma$, and we may leave out superscripts if the relevant wrapper w is clear from the context.

We remark that our theory does not depend on a specific choice of wrappers and approximations along them. Observation tables are just an instance. Other data structures representing approximations of automata can also be captured, as will be shown in Section 7.

Although the algebra f may be unknown—as for instance $f = \delta_M$ is unknown to automata learning algorithms—the approximated version ξ_f can sometimes be recovered via the following simple result, stating that approximations transfer along algebra homomorphisms.

▶ **Proposition 4.** *For all endofunctors F, F-algebras (U, u) and (V, v), F-algebra homomorphisms $h\colon (U, u) \to (V, v)$, and morphisms $\alpha\colon S \to U$ and $\beta\colon V \to P$, $\xi_v^{(h\circ\alpha,\beta)} = \xi_u^{(\alpha,\beta\circ h)}$.*

▶ **Example 5.** Consider again the wrapper in Example 2. There $\sigma = \mathsf{rch} \circ \mathsf{inc}$, where $\mathsf{inc}\colon S \to A^*$ is the inclusion of S into A^* and $\mathsf{rch}\colon A^* \to M$ is the full reachability map of M. If we equip A^* with transitions $\mathsf{cat}\colon A^* \times A \to A^*$ defined as $\mathsf{cat}(u, a) = ua$, then rch is an algebra homomorphism $(A^*, c) \to (M, \delta)$ for $(-) \times A$. We can therefore apply Proposition 4 to obtain $\xi_\delta^{(\mathsf{rch}\circ\mathsf{inc},\pi)} = \xi_c^{(\mathsf{inc},\pi\circ\mathsf{rch})}$. It follows that $\xi_\delta\colon S \times A \to 2^E$ is given by $\xi_\delta(s, a)(e) = \mathcal{L}(sae)$, which corresponds to the $S \cdot A \to 2^E$ part of the row function in Section 2.

3.2 Hypotheses as Factorizations

We now formalize the process of deriving a hypothesis automaton from an observation table. Recall from Section 2 that the state space of the hypothesis automaton is precisely the image of the component $S \to 2^E$ of the row function. Image factorizations are abstractly captured by the notion of $(\mathcal{E}, \mathcal{M})$ *factorization system* on a category. We will assume throughout the paper that our category $\mathbf{C}$ has $(\mathcal{E}, \mathcal{M})$ factorizations.

▶ **Definition 6** (Factorization System). A *factorization system* is a pair $(\mathcal{E}, \mathcal{M})$ of sets of morphisms such that

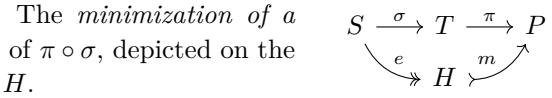
1. all morphisms f can be decomposed as $f = m \circ e$, with $m \in \mathcal{M}$ and $e \in \mathcal{E}$;
2. for all commutative squares as on the right, where $i \in \mathcal{E}$ and $j \in \mathcal{M}$, there is a unique diagonal d making the triangles commute;

$$\begin{array}{ccc} U & \xrightarrow{i} & V \\ {\scriptstyle g}\downarrow & \overset{d}{\swarrow} & \downarrow{\scriptstyle h} \\ W & \xrightarrow{j} & X \end{array}$$

3. both $\mathcal{E}$ and $\mathcal{M}$ are both closed under composition and contain all isomorphisms;
4. every morphism in $\mathcal{E}$ is an epi, and every morphism in $\mathcal{M}$ is a mono.

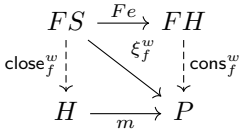
We will use double-headed arrows ($\twoheadrightarrow$) to indicate morphisms in $\mathcal{E}$ and arrows with a tail ($\rightarrowtail$) to indicate morphisms in $\mathcal{M}$. For instance, in **Set** the pair (surjective functions, injective functions) forms a factorization system, where each function $f\colon X \to Y$ can be decomposed as a surjective map $X \twoheadrightarrow \mathsf{img}(f)$ followed by the inclusion $\mathsf{img}(f) \rightarrowtail Y$.

▶ **Definition 7** (Wrapper Minimization)**.** The *minimization of a wrapper* (σ, π) is the $(\mathcal{E}, \mathcal{M})$-factorization of $\pi \circ \sigma$, depicted on the right. Notice that (e, m) is a wrapper for H.



In the wrapper of Example 2, H is the state space of the hypothesis automaton defined in Section 2. However, H does not yet have an automaton structure. In the concrete setting, this can be computed from the $S \cdot A \to 2^E$ part of row whenever closedness and consistency are satisfied. We give our abstract characterization of these requirements, which generalize the definitions due to Jacobs and Silva [19].

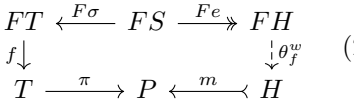
▶ **Definition 8** (Closedness and Consistency)**.** Let $w = (S \xrightarrow{\sigma} T, T \xrightarrow{\pi} P)$ be a wrapper and $(S \xrightarrow{e} H, H \xrightarrow{m} P)$ its minimization. Given an endofunctor F and a morphism $f\colon FT \to T$, we say that w is *f-closed* if there is a morphism $\mathsf{close}_f^w\colon FS \to H$ making the left triangle in (1) commute; we say that w is *f-consistent* if there is a morphism $\mathsf{cons}_f^w\colon FH \to P$ making the right triangle commute.



(1)

Closedness and consistency together yield an algebra structure on the hypothesis. This is not just any algebra, but one that relates the wrapper to its minimization, as we show next.

▶ **Theorem 9.** *Let $w = (S \xrightarrow{\sigma} T, T \xrightarrow{\pi} P)$ be a wrapper and $(S \xrightarrow{e} H, H \xrightarrow{m} P)$ its minimization. For each $\mathcal{E}$-preserving endofunctor F and each morphism $f\colon FT \to T$, w is both f-closed and f-consistent if and only if there exists $\theta_f^w\colon FH \to H$ making the diagram on the right commute.*

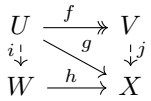


(2)

Proof. Given the morphisms close_f and cons_f, we let θ_f be the unique diagonal provided by the factorization system for the commutative square (1); conversely, given θ_f, we take $\mathsf{close}_f = \theta_f \circ Fe$ and $\mathsf{cons}_f = m \circ \theta_f$. ◀

To understand abstract closedness and consistency properties in **Set**, and to relate them to the concrete definitions given in Section 2, we give the following general result.

▶ **Proposition 10.** *Consider functions f, g, and h as in the diagram below, with f surjective.*



1. *A function $i\colon U \to W$ making the left triangle in the diagram commute exists if and only if for each $u \in U$ there is a $w \in W$ such that $h(w) = g(u)$.*

2. *A function $j\colon V \to X$ making the right triangle commute exists if and only if for all $u_1, u_2 \in U$ such that $f(u_1) = f(u_2)$ we have $g(u_1) = g(u_2)$.*

For the wrapper in Example 2 and $F = (-) \times A$, δ_M-closedness and δ_M-consistency coincide respectively with closedness and consistency as defined in Section 2. If these are satisfied, then the function θ_{δ_M} is precisely the transition function δ_H defined there. Notice that all endofunctors on **Set** preserve surjections.

Interestingly, closedness and consistency also tell us when initial and accepting states of H can be derived from the observation table. For initial states, we note that the initial state $m_0 \in M$ can be seen as a function $m_0\colon 1 \to M$. As the set 1 is the constant functor 1 applied to M, this initial state gives rise to another closedness property, which states that there must be an $s \in S$ such that $\xi(s)(e) = \mathcal{L}(e)$ for all $e \in E$. Note that m_0-consistency trivially holds because of the constant functor involved. When m_0-closedness is satisfied, $\mathsf{close}_{m_0}\colon 1 \to H$ is an initial state map. For instance, in $\mathtt{L}^\star$ this property is always satisfied because $\varepsilon \in S$.

For accepting states, one would expect a similar property regarding the set $F_M \subseteq M$, which can be represented by a function $F_M\colon M \to 2$. However, this is a coalgebra (for the constant functor 2) rather than an algebra. Fortunately, a wrapper $(S \xrightarrow{\sigma} T, T \xrightarrow{\pi} P)$ in $\mathbf{C}$ gives a wrapper (π, σ) in $\mathbf{C}^{\mathbf{op}}$, so a morphism $f\colon T \to FT$ in $\mathbf{C}$ yields the approximation $\xi_f^{(\pi,\sigma)} = S \xrightarrow{\sigma} T \xrightarrow{f} FT \xrightarrow{F\pi} FP$. In particular, for $F_M\colon M \to 2$, this leads to a consistency[1] property stating that for all $s_1, s_2 \in S$ such that $\xi(s_1) = \xi(s_2)$ we must have $\mathcal{L}(s_1) = \mathcal{L}(s_2)$. The $\mathtt{L}^\star$ algorithm ensures this by having $\varepsilon \in E$, using that $\mathcal{L}(s) = \xi(s)(\varepsilon)$ for any $s \in S$.

4 A General Correctness Theorem

In this section we work towards a general correctness theorem that is completed in Section 4.1, where we introduce an abstract notion of automaton. We then show in Section 4.2 how the theorem applies to the `ID` algorithm, to the algorithm by Arbib and Zeiger, and to $\mathtt{L}^\star$.

$$\begin{array}{ccc} S & \xrightarrow{\sigma} & T \\ {\scriptstyle e}\downarrow & \swarrow^{\phi} & \downarrow{\scriptstyle \pi} \\ H & \xrightarrow{m} & P \end{array} \qquad\qquad \begin{array}{ccc} S & \xrightarrow{e} & H \\ {\scriptstyle \sigma}\downarrow & \swarrow^{\psi} & \downarrow{\scriptstyle m} \\ T & \xrightarrow{\pi} & P \end{array} \tag{3}$$

The key observation for the correctness theorem is the following. Let $w = (S \xrightarrow{\sigma} T, T \xrightarrow{\pi} P)$ be a wrapper with minimization $(S \xrightarrow{e} H, H \xrightarrow{m} P)$. If $\sigma \in \mathcal{E}$, then the factorization system gives us a unique diagonal ϕ in the left square of (3), which by (the dual of) Proposition A.1 satisfies $\phi \in \mathcal{E}$. Similarly, if $\pi \in \mathcal{M}$, we have ψ in the right square of (3), with $\psi \in \mathcal{M}$. Composing the two diagrams and using again the unique diagonal property, one sees that ϕ and ψ must be mutually inverse. We can conclude that H and T are isomorphic. Now, if w is a wrapper produced by a learning algorithm and T is the state space of the target minimal automaton, as in Example 2, our reasoning hints at a correctness criterion: $\sigma \in \mathcal{E}$ and $\pi \in \mathcal{M}$ upon termination ensure that H is isomorphic to T. Of course, the criterion will have to guarantee that the automata, not just the state spaces, are isomorphic.

We first show that the argument above lifts to F-algebras $f\colon FT \to T$, for an arbitrary endofunctor $F\colon \mathbf{C} \to \mathbf{C}$ preserving $\mathcal{E}$.

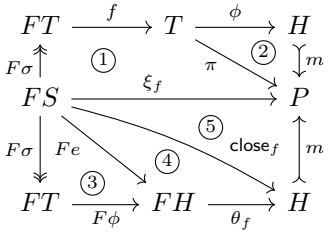
[1] Technically, this should be called coclosedness, as it is closedness in the category $\mathbf{C}^{\mathbf{op}}$. We choose to overload consistency so as not to obscure the terminology.

▶ **Lemma 11.** *For a wrapper $w = (\sigma, \pi)$ and an F-algebra f, if $\sigma \in \mathcal{E}$, then w is f-closed; if $\pi \in \mathcal{M}$, then w is f-consistent.*

Proof. If $\sigma \in \mathcal{E}$, then let ϕ be as in (3) and define $\mathsf{close}_f = \phi \circ f \circ F\sigma$; if $\pi \in \mathcal{M}$, then let ψ be as in (3) and define $\mathsf{cons}_f = \pi \circ f \circ F\psi$. ◀

▶ **Proposition 12.** *For a wrapper $w = (\sigma, \pi)$ and an F-algebra f, if $\sigma \in \mathcal{E}$ and w is f-consistent, then ϕ as given in (3) is an F-algebra homomorphism $(T, f) \to (H, \theta_f)$; if $\pi \in \mathcal{M}$ and w is f-closed, then ψ as given in (3) is an F-algebra homomorphism $(H, \theta_f) \to (T, f)$.*

Proof. Assume that $\sigma \in \mathcal{E}$ and w is f-consistent. The proof for the other part is analogous. Using Lemma 11 and Theorem 9 we see that θ_f exists. Since $F\sigma$ is epic and m monic, it suffices to show $m \circ \phi \circ f \circ F\sigma = m \circ \theta_f \circ F\phi \circ F\sigma$, which is done on the right.



① definition of ξ_f
② (3)
③ functoriality, (3)
④ definition of θ_f
⑤ closedness

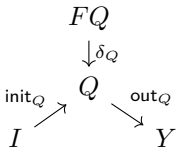
◀

▶ **Corollary 13.** *If $\sigma \in \mathcal{E}$ and $\pi \in \mathcal{M}$, then ϕ given in (3) is an F-algebra iso $(T, f) \to (H, \theta_f)$.*

4.1 Abstract Automata

Now we enrich F-algebras with initial and final states, obtaining a notion of *automaton* in a category. Then we give the full correctness theorem for automata. We fix objects I and Y in $\mathbf{C}$, which will serve as initial state selector and output of the automaton, respectively.

▶ **Definition 14** (Automaton)**.** An *automaton in* $\mathbf{C}$ is an object Q of $\mathbf{C}$ equipped with an *initial state map* $\mathsf{init}_Q \colon I \to Q$, an *output map* $\mathsf{out}_Q \colon Q \to Y$, and *dynamics* $\delta_Q \colon FQ \to Q$. An *input system* is an automaton without an output map; an *output system* is an automaton without an initial state map.



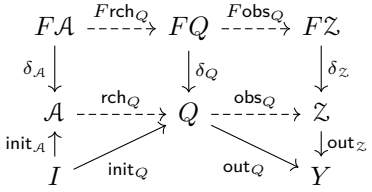
Automata form a category, where morphisms f between automata U and V are F-algebra homomorphisms that commute with initial state and output maps: $f \circ \mathsf{init}_U = \mathsf{init}_V$ and $\mathsf{out}_V \circ f = \mathsf{out}_U$. Composition and identities are as in $\mathbf{C}$. There are analogous categories of input and output systems.

To recover DAs (DFAs that may not be finite) over an alphabet A as automata, we take $I = 1$, $Y = 2$, and $F = (-) \times A$ in the category $\mathbf{C} = \mathbf{Set}$. An input system is then a DA without a classification of states into accepting and rejecting states; an output system is a DA without a designated initial state.

In Appendix B we explain how weighted automata can be modeled as automata. Nominal automata are automata in the category of nominal sets and equivariant functions if we take $I = 1$ with a trivial action, Y any nominal set, and $F = (-) \times A$ for a nominal set A. We do not treat this setting in the present paper, but the $\mathtt{L}^\star$ adaptation devised by Moerman et al. [24] agrees with our abstract definitions.

We assume the existence of an initial input system $\mathcal{A}$ and a final output system $\mathcal{Z}$. These give general notions of *reachability* and *observability*.

▶ **Definition 15** (Reachability and Observability)**.** The *reachability map* of an automaton Q is the unique input system homomorphism $\mathsf{rch}_Q \colon \mathcal{A} \to Q$; its *observability map* is the unique output system homomorphism $\mathsf{obs}_Q \colon Q \to \mathcal{Z}$. The automaton Q is *reachable* if $\mathsf{rch}_Q \in \mathcal{E}$; it is *observable* if $\mathsf{obs}_Q \in \mathcal{M}$. An automaton is *minimal* if it is both reachable and observable.



In the DA setting, the set of words A^* forms an initial input system. Its initial state is the empty word $\varepsilon \colon 1 \to A^*$, and its transition function $\mathsf{cat} \colon A^* \times A \to A^*$ is given by concatenation: $\mathsf{cat}(u, a) = ua$. This yields the expected definition of the reachability map $\mathsf{rch}_Q \colon A^* \to Q$ for a DA Q: $\mathsf{rch}_Q(\varepsilon) = \mathsf{init}_Q(*)$ and $\mathsf{rch}_Q(ua) = \delta_Q(\mathsf{rch}_Q(u), a)$. The set of languages 2^{A^*} in this setting forms a final output system. Its accepting states $\varepsilon? \colon 2^{A^*} \to 2$ are those languages that contain the empty word, $\varepsilon?(L) = L(\varepsilon)$, and its dynamics $\mathsf{deriv} \colon 2^{A^*} \times A \to 2^{A^*}$ is given by $\mathsf{deriv}(L, a)(v) = L(av)$. The observability map of a DA Q assigns to each state the language it accepts: $\mathsf{obs}_Q(q)(\varepsilon) = \mathsf{out}_Q(q)$ and $\mathsf{obs}_Q(q)(av) = \mathsf{obs}_Q(\delta_Q(q, a))(v)$.

In general, we define the *language* of an automaton Q to be $\mathcal{L}_Q = \mathsf{out}_Q \circ \mathsf{rch}_Q \colon \mathcal{A} \to Y$. For DFAs, this is the usual definition of the accepted language $A^* \to 2$ (alternatively, $\mathcal{L}_Q = \mathsf{obs}_Q \circ \mathsf{init}_Q \colon 1 \to 2^{A^*}$, which explicitly mentions the initial state). If for automata U and V there exists an automaton homomorphism $U \to V$, one can prove that $\mathcal{L}_U = \mathcal{L}_V$.

We are now ready to give the main result of this section, which extends Proposition 12. Intuitively, it provides conditions for when the hypothesis automaton is actually the target automaton. Remarkably, unlike Proposition 12, it is enough to require just one of $\sigma \in \mathcal{E}$ and $\pi \in \mathcal{M}$, thanks to observability and reachability.

▶ **Theorem 16.** *Let $w = (S \xrightarrow{\sigma} Q, Q \xrightarrow{\pi} P)$ be a wrapper for an automaton Q and let H be the hypothesis automaton for w. If at least one the following is true:*
1. *Q is observable and w is an out_Q-consistent and δ_Q-consistent wrapper such that $\sigma \in \mathcal{E}$;*
2. *Q is reachable and w is an init_Q-closed and δ_Q-closed wrapper such that $\pi \in \mathcal{M}$;*
then H and Q are isomorphic automata.

Proof. We only show point 1; the other is analogous. Recall that the initial state map of the automaton is an algebra for the constant functor I while the output map is a coalgebra for the constant functor Y. Hence, we can apply Proposition 12 (or its dual for the coalgebra) to them and to δ_Q to find that $\phi \colon Q \to H$ is an automaton homomorphism. Then $\mathsf{obs}_Q = \mathsf{obs}_H \circ \phi$ by finality of $\mathcal{Z}$. Since obs_Q is in $\mathcal{M}$, this means that $\phi \in \mathcal{M}$ (see Proposition A.1). Because $m \in \mathcal{M}$ and $\pi = m \circ \phi$ (3), we have $\pi \in \mathcal{M}$. Therefore, we can apply Corollary 13, again three times, and obtain an automaton isomorphism between Q and H. ◀

4.2 Applications to Known Learning Algorithms

Recall that `ID` assumes a finite set $S \subseteq A^*$ such that each state of the minimal target DFA M is reached by reading one of the words in S when starting from m_0. In the terminology of the previous section, the algorithm takes a wrapper $(S \xrightarrow{\sigma} M, M \xrightarrow{\pi} 2^E)$ as in Example 2, with E initialized to $\{\varepsilon\}$, extends E for δ-consistency, and obtains M as the hypothesis.

The correctness of the algorithm can be explained via Theorem 16(1) as follows. The assumption of `ID` about S is equivalent to $\sigma \in \mathcal{E}$, because σ is defined to be the reachability

map restricted to S. The out-consistency condition is satisfied by initializing $E = \{\varepsilon\}$, and ensuring δ-consistency is precisely what the algorithm does. Therefore, by Theorem 16(1), the final wrapper yields a hypothesis DFA isomorphic to M.

Theorem 16(2) suggests a dual to this algorithm, where we assume a finite set $E \subseteq A^*$ such that every pair of different states of M is distinguished by some word in E. Enforcing closedness will lead to a hypothesis DFA isomorphic to M.

We can also explain the Arbib and Zeiger algorithm. In a minimal DFA M with at most n states, every state is reached via a word of length at most $n-1$; similarly, every pair of states is distinguished by a word of length up to $n-1$. The algorithm of Arbib and Zeiger takes a wrapper $(S \xrightarrow{\sigma} M, M \xrightarrow{\pi} 2^E)$ as in Example 2, with $S = E = A^{\leq n-1}$, and by Corollary 13 (applied once for each of init_M, δ_M, and out_M) immediately obtains M up to isomorphism as the hypothesis. We note that taking this large E is unnecessary, as we could simply apply Theorem 16(1), thus reducing the algorithm to $\mathtt{ID}$.

Finally, we consider the $\mathtt{L}^\star$ algorithm. Let the wrapper $(S \xrightarrow{\mathsf{inc}} A^* \xrightarrow{\mathsf{rch}} M, M \xrightarrow{\pi} 2^E)$ be as in Example 2, and let H be its hypothesis DFA. We have the following result.

▶ **Proposition 17.** *If for every prefix p of a word $z \in A^*$ there exists an $s \in S$ such that $\mathsf{rch}_M(s) = \mathsf{rch}_M(p)$, then $\mathcal{L}_H(z) = \mathcal{L}_M(z)$.*

▶ **Corollary 18.** *If $z \in A^*$ is a counterexample, i.e., $\mathcal{L}_H(z) \neq \mathcal{L}_M(z)$, then adding all prefixes of z to S will increase the size of $\mathsf{img}(\mathsf{rch}_M \circ \mathsf{inc})$.*

Thus, after $\mathtt{L}^\star$ has processed at most $|M|$ counterexamples, the conditions for Theorem 16(1) are satisfied, which means that the next hypothesis will be isomorphic to M.

5 Minimization

In this section we explore the connection (1) in Figure 1 between automata learning and minimization algorithms. Recall that minimization algorithms typically have two phases: a reachability analysis phase, which removes unreachable states; and a merging phase, where language-equivalent states are merged. We will rephrase these two phases in the terminology of Section 2, and we will show that Theorem 16 can be used to explain their correctness. We fix a DFA Q throughout the section.

Reachability Analysis Phase. Let R be the reachable part of Q and S any subset of R. There is a wrapper $w = (S \xrightarrow{\sigma} R, R \xrightarrow{\pi} Q)$, where σ and π are just the inclusions. The set S models the state of an algorithm performing a reachability analysis on Q. Since σ and π are inclusions, the hypothesis for w is the set S itself.

As the inclusion π is an automaton homomorphism, by Proposition 4 we can use the transition function $\delta\colon Q \times A \to Q$ and initial state $q_0\colon 1 \to Q$ to compute

$$\xi^{(\sigma,\pi)}_{\delta\colon R\times A\to R} = \xi^{(\pi\circ\sigma,\mathsf{id})}_{\delta\colon Q\times A\to Q}\colon S \times A \to Q \qquad \xi^{(\sigma,\pi)}_{r_0\colon 1\to R} = \xi^{(\pi\circ\sigma,\mathsf{id})}_{q_0\colon 1\to Q} = q_0\colon 1 \to Q.$$

The function ξ_δ simply assigns to each state $s \in S$ and each symbol $a \in A$ the state $\delta_Q(s,a)$. The wrapper is therefore δ_R-closed if for all $q \in S$ and $a \in A$ we have $\delta_Q(q,a) \in S$; it is r_0-closed if $q_0 \in S$. The obvious algorithm to ensure these closedness properties, namely initializing $S = \{q_0\}$ and adding $\delta_Q(q,a)$ to S while there are $q \in S$ and $a \in A$ such that $\delta_Q(q,a) \notin S$, is the usual reachability analysis algorithm. Since R is reachable and π injective, Theorem 16(2) confirms that this algorithm finds an automaton isomorphic to R.

Alternatively, one could let S be a subset of A^* and use the wrapper $(S \xrightarrow{\mathsf{inc}} A^* \xrightarrow{\mathsf{rch}} R, A^* \xrightarrow{\pi} Q)$, where inc is the inclusion and rch the reachability map for R (see Definition 15).

Starting from $S = \{\varepsilon\}$, the algorithm would add to S words reaching states not yet visited. This is closer to how automata learning algorithms fix closedness.

State Merging Phase. Now we are interested in finding the DFA O that is obtained from Q by merging states that accept the same language. Formally, this automaton is obtained by factorizing the observability map obs_Q for Q (see Definition 15) as DA homomorphisms $Q \overset{h}{\twoheadrightarrow} O \overset{\mathsf{obs}}{\rightarrowtail} 2^{A^*}$. Given a finite set $E \subseteq A^*$, these can be made into a wrapper $w = (Q \overset{h}{\to} O, O \xrightarrow{\mathsf{obs}} 2^{A^*} \xrightarrow{\mathsf{res}} 2^E)$, where res restricts a language to the words in E. Consider $\xi^w \colon Q \to 2^E$ (i.e., the composition of the morphisms in w): even though O is not known a priori, this function can be computed by testing for a given state which words in E it accepts. Because h is an automaton homomorphism, $\xi^w_{\delta_O}$ by Proposition 4 equals $\xi^{(\mathsf{id},\xi^w)}_{\delta_Q} = \xi^w \circ \delta_Q$. Since h is surjective, Theorem 16(1) says that we only have to ensure δ-consistency and out-consistency. One may start with $E = \{\varepsilon\}$ to satisfy the latter. For δ-consistency, for all $q_1, q_2 \in Q$ such that $\xi(q_1) = \xi(q_2)$ we must have $\xi(\delta(q_1, a)) = \xi(\delta(q_2, a))$ for each $a \in A$. This can be ensured in the same fashion as for an observation table.

The algorithm we have just described reminds of Moore's reduction procedure [25]. However, using a table as data structure is less efficient because many cells may be redundant. The same redundancy has been observed in the $\mathrm{L}^\star$ algorithm. In Section 7 we will show how CALF covers optimized versions of these algorithms, but first we discuss yet another application of our framework.

6 Conformance Testing

In this section we consider the connections ② and ③ in Figure 1 between testing and, respectively, automata learning and minimization. More precisely, we consider an instance of *conformance testing* where a known DFA U is tested for equivalence against a black-box DFA V. One application of this problem is the realization of equivalence queries in the $\mathrm{L}^\star$ algorithm, where U is the hypothesis DFA and V is the target.

Testing consists in comparing U and V on a finite set of words, approximating their behavior. Following the idea of using wrappers to generalize approximations, we now capture testing using our abstract learning machinery. This will allows us to explain correctness of testing using Theorem 16. First note that given objects S and P and morphisms $\alpha \colon S \to \mathcal{A}$ and $\omega \colon \mathcal{Z} \to P$, we can associate wrappers to both the known automaton U and the black-box V by composing with their reachability and observability maps:

$$w_U = (\sigma_U, \pi_U) \qquad \sigma_U = S \xrightarrow{\alpha} \mathcal{A} \xrightarrow{\mathsf{rch}_U} U \qquad \pi_U = U \xrightarrow{\mathsf{obs}_U} \mathcal{Z} \xrightarrow{\omega} P$$
$$w_V = (\sigma_V, \pi_V) \qquad \sigma_V = S \xrightarrow{\alpha} \mathcal{A} \xrightarrow{\mathsf{rch}_V} V \qquad \pi_V = V \xrightarrow{\mathsf{obs}_V} \mathcal{Z} \xrightarrow{\omega} P.$$

We now use several approximations in defining the *tests of U against V*, covering transition functions, initial states, and final states:

$$\xi^{w_U} = \xi^{w_V} \qquad \xi^{w_U}_{\mathsf{init}_U} = \xi^{w_V}_{\mathsf{init}_V} \qquad \xi^{w_U}_{\delta_U} = \xi^{w_V}_{\delta_V} \qquad \xi^{w_U}_{\mathsf{out}_U} = \xi^{w_V}_{\mathsf{out}_V}. \tag{4}$$

▶ **Example 19.** If U and V are DFAs, we can choose wrappers as in Example 2, namely by letting $\alpha = \mathsf{inc} \colon S \to A^*$ be an inclusion and $\omega = \mathsf{res} \colon 2^{A^*} \to 2^E$ a restriction. The approximations along w_U are then given by $\xi^{w_U}(s)(e) = \mathcal{L}_U(se)$, $\xi^{w_U}_{\mathsf{init}_U}(e) = \mathcal{L}_U(e)$, $\xi^{w_U}_{\delta_U}(s, a)(e) = \mathcal{L}_U(sae)$, and $\xi^{w_U}_{\mathsf{out}_U}(s) = \mathcal{L}_U(s)$, and analogously for w_V. Thus, checking (4) here amounts to checking whether the DFAs accept exactly the same words from the set $S \cdot E \cup E \cup S \cdot A \cdot E \cup S$.

Our main result regarding conformance testing captures the properties of the wrappers that need to hold for the above tests to prove that U is equivalent to the black-box V.

▶ **Theorem 20.** *Given the above wrappers, suppose $\sigma_U \in \mathcal{E}$, $\pi_U \in \mathcal{M}$, and either $\sigma_V \in \mathcal{E}$ and V is observable or $\pi_V \in \mathcal{M}$ and V is reachable. Then U is isomorphic to V if and only if all the equalities (4) hold.*

We now comment on the connection between testing and minimization algorithms. Minimization is a natural choice when looking for a set of words approximating U to be tested against V. Formally, recall from Section 5 that we can use reachability and state merging to find sets $S, E \subseteq A^*$. Moreover, reachability analysis gives an $\mathsf{inc}\colon S \to A^*$ that makes σ_U surjective and state merging gives a $\mathsf{res}\colon 2^{A^*} \to 2^E$ that makes π_U injective (recall that $\mathcal{A} = A^*$ and $\mathcal{Z} = 2^{A^*}$ in the DFA setting). These, together with reachability and observability maps, give wrappers for U and V. The condition of Theorem 20 on w_V may not hold right away, but these wrappers are convenient starting points for algorithmic techniques, as we now show.

W-method. We instantiate the above framework to recover the W-method [14]. This algorithm assumes to be given an upper bound n on the number of states of the unknown DFA V. Assume for convenience that U and V are minimal DFAs. We apply our framework as follows: first, we build the wrapper $w_U = (\sigma_U, \pi_U)$. We use the minimization algorithms from Section 5 to find $S \xrightarrow{\mathsf{inc}} A^*$ and $2^{A^*} \xrightarrow{\mathsf{res}} 2^E$ with finite $S, E \subseteq A^*$, which yield $\sigma_U = \mathsf{rch}_U \circ \mathsf{inc} \in \mathcal{E}$ and $\pi_U = \mathsf{res} \circ \mathsf{obs}_U \in \mathcal{M}$. If at this point the equalities (4) do not hold, then we can conclude that U and V accept different languages, and the testing failed. If we assume they hold, then because $\sigma_U \in \mathcal{E}$ and $\pi_U \in \mathcal{M}$ this means that $|\mathsf{img}(\xi^{w_V})| = |\mathsf{img}(\xi^{w_U})| = |U|$. The image of $\xi^{w_V} = \pi_V \circ \sigma_V$ is at least as big as the image of σ_V, so $|\mathsf{img}(\sigma_V)| \geq |U|$. By assumption we know that the size of V is at most n, and hence we update S to $S \cdot A^{\leq n-|U|}$, which yields $\sigma_V \in \mathcal{E}$ because $\varepsilon \in S$. Now we have $\sigma_U \in \mathcal{E}$, $\pi_U \in \mathcal{M}$, and $\sigma_V \in \mathcal{E}$. Applying Theorem 20, we can find out whether U and V are isomorphic by testing (4) for the updated wrappers. Instantiating what the equalities in (4) mean (see Example 19), we recover the test sequences generated by the W-method.

7 Optimized Algorithms

When fixing a consistency defect in an observation table, we add a column to distinguish two currently equal rows. Unfortunately, adding a full column implies that a membership query is needed also for each other row. Kearns and Vazirani [20] avoid this redundancy by basing the learning algorithm on a *classification tree* rather than an observation table. We now show that CALF encompasses this optimized algorithm, which will allow us to derive optimizations for other algorithms. First we introduce a notion of *classification tree* (also called *discrimination tree*), close to the one by Isberner [18].

▶ **Definition 21** (Classification Tree)**.** Given a finite $S \subseteq A^*$, a *classification tree* τ on S is a labeled binary tree with internal nodes labeled by words from A^* and leaves labeled by subsets of S.

The *language classifier* for τ is the function $\omega_\tau\colon 2^{A^*} \to \mathcal{P}S$ that operates as follows. Given a language $L \in 2^{A^*}$, starting from the root of the tree: if the current node is a leaf with label $U \in \mathcal{P}S$, the function returns U; if the current node is an internal node with label

$v \in A^*$, we repeat the process from the left subtree if $v \in L$ and from the right subtree otherwise.

In the CALF terminology, each classification tree τ gives rise to a wrapper ($S \xrightarrow{\mathsf{inc}} A^* \xrightarrow{\mathsf{rch}} M, M \xrightarrow{\mathsf{obs}} 2^{A^*} \xrightarrow{\omega_\tau} \mathcal{P}S$), where inc is the inclusion and M is the minimal DFA for the language $\mathcal{L}$ that is to be learned. We define a function $\mathsf{sift}_\tau \colon A^* \to \mathcal{P}S$ as the composition $\omega_\tau \circ \mathsf{obs}_M \circ \mathsf{rch}_M$. This function *sifts* [20] words $u \in A^*$ through the tree by moving on a node with label $v \in A^*$ to the subtree corresponding to the value of $\mathcal{L}(uv)$. Applying Proposition 4, the approximated initial state map and dynamics of M can be rewritten using this function: $\xi_{\mathsf{init}} = \mathsf{sift}_\tau \circ \varepsilon \colon 1 \to \mathcal{P}S$ and $\xi_\delta = \mathsf{sift}_\tau \circ \mathsf{cat} \circ (\mathsf{inc} \times \mathsf{id}_A) \colon S \times A \to \mathcal{P}S$. Recall that $\varepsilon \colon 1 \to A^*$ is the empty word and $\mathsf{cat} \colon A^* \times A \to A^*$ concatenates its arguments. The function $\xi_{\mathsf{out}} \colon S \to 2$ is still the same as for an observation table: $\xi_{\mathsf{out}}(s) = \mathcal{L}(s)$. From these definitions we obtain the following notions of closedness and consistency. The wrapper is:

- δ-closed if for each $t \in S \cdot A$ there is an $s \in S$ such that $\mathsf{sift}_\tau(s) = \mathsf{sift}_\tau(t)$;
- δ-consistent if for all $s_1, s_2 \in S$ such that $\mathsf{sift}_\tau(s_1) = \mathsf{sift}_\tau(s_2)$ we have $\mathsf{sift}_\tau(s_1 a) = \mathsf{sift}_\tau(s_2 a)$ for any $a \in A$;
- init-closed if there is an $s \in S$ such that $\mathsf{sift}_\tau(s) = \mathsf{sift}_\tau(\varepsilon)$;
- out-consistent if for all $s_1, s_2 \in S$ such that $\mathsf{sift}_\tau(s_1) = \mathsf{sift}_\tau(s_2)$ we have $\mathcal{L}(s_1) = \mathcal{L}(s_2)$.

Now we can optimize the learning algorithm using Kearns and Vazirani classification trees as follows. Initially, the tree is just a leaf containing all words in S, which may be initialized to $\{\varepsilon\}$. When an out-consistency or δ-consistency defect is found, we have two words $s_1, s_2 \in S$ such that $\xi(s_1) = \xi(s_2) = U$, for some $U \in \mathcal{P}S$. We also have a word $v \in A^*$ such that $\mathcal{L}(s_1 v) \neq \mathcal{L}(s_2 v)$,[2] and we want to use this word to update the tree τ to distinguish $\xi(s_1)$ and $\xi(s_2)$. This is done by replacing the leaf with label U by a node that distinguishes based on the word v. Its left subtree is a leaf containing the words $s \in U$ such that $\mathcal{L}(sv) = 0$ while the $s \in U$ in its right subtree are such that $\mathcal{L}(sv) = 1$. This requires new membership queries, but only one for each word in $S \cup S \cdot A$ that sifts into the leaf with label U; the observation table approach needs queries for all elements of $S \cup S \cdot A$ when a column is added.

The intention of having $\mathcal{P}S$ as the set of labels is that we maintain the trees in such a way that $\xi \colon S \to \mathcal{P}S$ maps each word in S to the unique leaf it is contained in. As a result, the function ξ can be read directly from the tree. This means that, when adding a word to S, we need to add it also to the leaf of the tree that it sifts into. Words are added to S when processing a counterexample as in $\mathtt{L}^\star$ and when fixing init-closedness or δ-closedness. These closedness defects occur when a word in $\{\varepsilon\} \cup S \cdot A$ sifts into an empty set leaf. In that case we add the word to S.

Classification trees were originally developed for $\mathtt{L}^\star$, but we note that they can be used in `ID` as well. By the abstract nature of Theorem 16, no new correctness proof is necessary.

Transporting Optimizations to Minimization and Testing. Using our correspondence between learning and minimization, the above optimization for learning algorithms immediately inspires an optimization for the state merging phase of Section 5. The main difference is that we sift states of the automaton Q through the tree, rather than words. That is, when sifting a state q at a node with label $v \in A^*$, the subtree we move to corresponds to whether v is accepted by q. Thus, instead of taking the labels of leaves from $\mathcal{P}S$, we take them from $\mathcal{P}Q$. The algorithm described above now creates a *splitting tree* [21, 28] for Q.

[2] For an out-inconsistency, $v = \varepsilon$; on a δ-inconsistency, there is an $a \in A$ such that $\mathsf{sift}_\tau(s_1 a) \neq \mathsf{sift}_\tau(s_2 a)$. We take the label u of the lowest common ancestor node of those two leaves and define $v = au$.

Interestingly, in this case one does not actually have to represent the trees to perform the algorithm; tracking only the partitioning of Q induced by the tree is enough. This is because the classification of next states is contained in the classification of states: $\xi_\delta = \xi \circ \delta$ (using Proposition 4). An inconsistency consists in two states being in the same partition, but for some input $a \in A$ the corresponding next states being in different partitions. One then splits the partition of the two states into one partition for each of the partitions obtained after reading a. This corresponds to (repeatedly[3]) updating the tree to split the leaf. This algorithm that does not keep track of the tree is precisely Moore's reduction procedure [25].

The splitting tree algorithm described above can also be plugged into the W-method as it is described in Section 6. Note that the discussion about the correctness of the testing algorithm is not affected by this. The resulting algorithm is closely related to the HSI-method [22].

8 Conclusion

We have presented CALF, a categorical automata learning framework. CALF covers the basic definitions used in automata learning algorithms and unifies correctness proofs for several of them. We have shown that these proofs extend also to minimization and testing methods. CALF is general enough to capture optimizations for all of these algorithms and provides an abstract umbrella to transfer results between the three areas. We illustrated how an optimization known in learning can be transported to minimization and testing. We have also exploited the categorical nature of the framework by changing the category and deriving algorithms for weighted automata in Appendix B. This example shows the versatility of the framework: dynamics of weighted automata are naturally presented as coalgebras, and CALF can accommodate this perspective as well as the algebraic one, which is used traditionally for DFAs and adopted in the main text.

Related Work. Most of the present paper is based on the first author's Master's thesis [29].

Other frameworks have been developed to study automata learning algorithms, e.g. [6, 18]. However, in both cases the results are much less general than ours and not applicable for example to settings where the automata have additional structure, such as weighted automata (Appendix B). Isberner [18] hints at the connection between algorithms for minimization and learning. For example, before introducing classification trees for learning, he explains them for minimization. Still, he does not provide a formal link between the two algorithms.

The relation between learning and testing was first explored by Berg et al. [7]. Their discussion, however, is limited to the case where the black-box DFA has at most as many states as the known DFA. This is because their correspondence is a stronger one that relates terminated learning algorithms to test sets of conformance testing algorithms, whereas we have provided algorithmic connections. Our Theorem 20, which allows reasoning about algorithms not making the above assumption, is completely new.

As mentioned in the introduction, a preliminary investigation of generalizing automata learning concepts using category theory was performed by Jacobs and Silva [19]. We were inspired by their work, but we note that they did not attempt to formulate anything as general as our wrappers; definitions are concrete and dependent on observation tables. As a result, there are no fully abstract proofs and no instantiations to optimizations such as the classification trees discussed in Section 7. Jacobs and Silva also investigated weighted

[3] The partition splitting algorithm resolves every inconsistency for a within the partition at once.

automata, as we do in Appendix B, but only for the $L^\star$ algorithm. We note that Bergadano and Varricchio [8] first adapted $L^\star$ and ID for weighted automata.

Future Work. Many directions for future research are left open. An interesting idea inspired by the relation between testing and learning is integrating testing algorithms into $L^\star$. This could lead to optimizations that are unavailable when those components are kept separate. Further additions to CALF may include an investigation of the minimality of hypotheses and a characterization of the more elementary steps that are used in the algorithms. The only fully abstract correctness proofs for concrete algorithms at the moment are for the minimization algorithms and ID, where correctness follows from a condition that can be formulated on an abstract level. We would like to have an abstract characterization of progress for the other algorithms, which are of a more iterative nature. Finally, a concrete topic is optimizations for automata with additional structure, such as nondeterministic, weighted, and nominal automata. To the best of our knowledge, no analogue of classification trees exists for learning these classes. If such analogues turn out to exist, then the correspondences discussed in Section 5 and Section 6 would provide optimized learning and testing algorithms as well.

It is our long-term goal to exploit the practical aspects of the framework. For more details, see our project website `http://calf-project.org`.

Acknowledgements. We thank Joshua Moerman and a reviewer for useful comments.

References

1 Dana Angluin. A note on the number of queries needed to identify regular languages. *Inform. Control*, 51:76–87, 1981. doi:10.1016/S0019-9958(81)90090-5.

2 Dana Angluin. Learning regular sets from queries and counterexamples. *Inform. Comput.*, 75:87–106, 1987. doi:10.1016/0890-5401(87)90052-6.

3 Dana Angluin, Sarah Eisenstat, and Dana Fisman. Learning regular languages via alternating automata. In *IJCAI*, pages 3308–3314, 2015.

4 Dana Angluin and Dana Fisman. Learning regular omega languages. In *ALT*, volume 8776, pages 125–139, 2014. doi:10.1007/978-3-319-11662-4_10.

5 Michael A. Arbib and H. Paul Zeiger. On the relevance of abstract algebra to control theory. *Automatica*, 5:589–606, 1969. doi:10.1016/0005-1098(69)90026-0.

6 José L. Balcázar, Josep Díaz, Ricard Gavaldà, and Osamu Watanabe. Algorithms for learning finite automata from queries: A unified view. In *Advances in Algorithms, Languages, and Complexity*, pages 53–72. Springer, 1997.

7 Therese Berg, Olga Grinchtein, Bengt Jonsson, Martin Leucker, Harald Raffelt, and Bernhard Steffen. On the correspondence between conformance testing and regular inference. In *FASE*, volume 3442, pages 175–189, 2005. doi:10.1007/978-3-540-31984-9_14.

8 Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM Journal on Computing*, 25(6):1268–1280, 1996. doi:10.1137/S009753979326091X.

9 Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In *IJCAI*, volume 9, pages 1004–1009, 2009.

10 Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. A fresh approach to learning register automata. In *DLT*, volume 7907, pages 118–130, 2013. doi:10.1007/978-3-642-38771-5_12.

11 Sofia Cassel, Falk Howar, Bengt Jonsson, and Bernhard Steffen. Active learning for extended finite state machines. *FAC*, 28(2):233–263, 2016. doi:10.1007/s00165-016-0355-5.

12 Martin Chapman, Hana Chockler, Pascal Kesseli, Daniel Kroening, Ofer Strichman, and Michael Tautschnig. Learning the language of error. In *ATVA*, volume 9364, pages 114–130. Springer, 2015. doi:10.1007/978-3-319-24953-7_9.

13 Chia Yuan Cho, Domagoj Babić, Eui Chul Richard Shin, and Dawn Song. Inference and analysis of formal models of botnet command and control protocols. In *CCS*, pages 426–439. ACM, 2010. doi:10.1145/1866307.1866355.

14 Tsun S. Chow. Testing software design modeled by finite-state machines. *IEEE Trans. Software Eng.*, 4:178–187, 1978. doi:10.1109/TSE.1978.231496.

15 Joeri de Ruiter and Erik Poll. Protocol state fuzzing of TLS implementations. In *USENIX Security*, pages 193–206, 2015.

16 E. Mark Gold. System identification via state characterization. *Automatica*, 8:621–636, 1972.

17 Bin-Lun Ho. *On effective construction of realizations from input-output descriptions*. PhD thesis, Stanford University, 1966.

18 Malte Isberner. *Foundations of active automata learning: an algorithmic perspective*. PhD thesis, Technical University of Dortmund, 2015.

19 Bart Jacobs and Alexandra Silva. Automata learning: A categorical perspective. In *Horizons of the Mind*, volume 8464, pages 384–406, 2014. doi:10.1007/978-3-319-06880-0_20.

20 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT press, 1994.

21 David Lee and Mihalis Yannakakis. Testing finite-state machines: State identification and verification. *IEEE T. Comput.*, 43:306–320, 1994. doi:10.1109/12.272431.

22 Gang Luo, Alexandre Petrenko, and Gregor v. Bochmann. Selecting test sequences for partially-specified nondeterministic finite state machines. In *Protocol Test Systems*, pages 95–110. Springer, 1995.

23 Oded Maler and Amir Pnueli. On the learnability of infinitary regular sets. *Inform. and Comput.*, 118:316–326, 1995. doi:10.1006/inco.1995.1070.

24 Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michał Szynwelski. Learning nominal automata. In *POPL*, pages 613–625, 2017.

25 Edward F. Moore. Gedanken-experiments on sequential machines. *Automata studies*, 34:129–153, 1956.

26 Anil Nerode. Linear automaton transformations. *Proceedings of the AMS*, 9:541–544, 1958.

27 Mathijs Schuts, Jozef Hooman, and Frits Vaandrager. Refactoring of legacy software using model learning and equivalence checking: an industrial experience report. In *IFM*, volume 9681, pages 311–325, 2016. doi:10.1007/978-3-319-33693-0_20.

28 Rick Smetsers, Joshua Moerman, and David N. Jansen. Minimal separating sequences for all pairs of states. In *LATA*, pages 181–193, 2016. doi:10.1007/978-3-319-30000-9_14.

29 Gerco van Heerdt. An abstract automata learning framework. Master's thesis, Radboud University Nijmegen, 2016.

A Omitted Proofs

▶ **Proposition 4.** *For all endofunctors F, F-algebras (U,u) and (V,v), F-algebra homomorphisms $h\colon (U,u) \to (V,v)$, and morphisms $\alpha\colon S \to U$ and $\beta\colon V \to P$, $\xi_v^{(h\circ\alpha,\beta)} = \xi_u^{(\alpha,\beta\circ h)}$.*

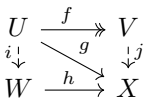
Proof. The F-algebra homomorphism $h\colon (U,u) \to (V,v)$ satisfies $v \circ Fh = h \circ u$ by definition.

Therefore,

$$\begin{aligned}
\xi_v^{(h\circ\alpha,\beta)} &= \beta \circ v \circ F(h \circ \alpha) && \text{(definition of } \xi_v^{(h\circ\alpha,\beta)}\text{)} \\
&= \beta \circ v \circ Fh \circ F\alpha && \text{(functoriality of } F\text{)} \\
&= \beta \circ h \circ u \circ F\alpha && (v \circ Fh = h \circ u) \\
&= \xi_u^{(\alpha,\beta\circ h)} && \text{(definition of } \xi_u^{(\alpha,\beta\circ h)}\text{).}
\end{aligned}$$

◀

▶ **Proposition 10.** *Consider functions f, g, and h as in the diagram below, with f surjective.*



1. *A function $i\colon U \to W$ making the left triangle in the diagram commute exists if and only if for each $u \in U$ there is a $w \in W$ such that $h(w) = g(u)$.*
2. *A function $j\colon V \to X$ making the right triangle commute exists if and only if for all $u_1, u_2 \in U$ such that $f(u_1) = f(u_2)$ we have $g(u_1) = g(u_2)$.*

Proof.

1. If such an i exists, then for each $u \in U$ we have $h(i(u)) = g(u)$. Conversely, define $i(u)$ to be any $w \in W$ such that $h(w) = g(u)$, which exists by assumption. Then $(h \circ i)(u) = h(w) = g(u)$.
2. If such a j exists, then whenever $f(u_1) = f(u_2)$ we have $(j \circ f)(u_1) = (j \circ f)(u_2)$ and therefore $g(u_1) = g(u_2)$. Conversely, define $j(f(u))$ to be $g(u)$, using that f is surjective. We only need to check that this is well-defined; i.e., that whenever $f(u_1) = f(u_2)$ we also have $g(u_1) = g(u_2)$. This is precisely the assumption. ◀

▶ **Proposition 17.** *If for every prefix p of a word $z \in A^*$ there exists an $s \in S$ such that $\mathsf{rch}_M(s) = \mathsf{rch}_M(p)$, then $\mathcal{L}_H(z) = \mathcal{L}_M(z)$.*

Proof. Let $(S \xrightarrow{\mathsf{inc}} A^* \xrightarrow{\mathsf{rch}} M, M \xrightarrow{\pi} P)$ be the wrapper and $(S \xrightarrow{e} H, H \xrightarrow{m} 2^E)$ its minimization. Furthermore, let $z = a_1 a_2 \dots a_n$ for n the length of z and each $a_i \in A$. For $0 \leq i \leq n$, define $z_i = a_1 a_2 \dots a_n$. Each prefix of z is z_i for some i. Assume that for every i there is an $s_i \in S$ such that $\mathsf{rch}_M(s_i) = \mathsf{rch}_M(z_i)$. We will show by induction to n that for all i,

$$\xi(s_i) = (m \circ \mathsf{rch}_H)(z_i). \tag{5}$$

Note that $z_0 = \varepsilon$. We have

$$\begin{aligned}
\xi(s_0) &= (\pi \circ \mathsf{rch}_M)(s_0) && \text{(definition of } \xi\text{)} \\
&= (\pi \circ \mathsf{rch}_M)(\varepsilon) && (\mathsf{rch}_M(s_0) = \mathsf{rch}_M(z_0)) \\
&= (\pi \circ \mathsf{init}_M)(*) && \text{(definition of } \mathsf{rch}_M\text{)} \\
&= \xi_{\mathsf{init}}(*) && \text{(definition of } \xi_{\mathsf{init}}\text{)} \\
&= (m \circ \mathsf{init}_H)(*) && \text{(definition of } \mathsf{init}_H\text{)} \\
&= (m \circ \mathsf{rch}_H)(\varepsilon) && \text{(definition of } \mathsf{rch}_H\text{).}
\end{aligned}$$

Now assume that for a certain $1 \leq i < n$ we have (5). This implies $e(s_i) = \mathsf{rch}_H(z_i)$ because $\xi = m \circ e$ and m is injective. Then

$$\begin{aligned}
\xi(s_{i+1}) &= (\pi \circ \mathsf{rch}_M)(s_{i+1}) && \text{(definition of } \xi) \\
&= (\pi \circ \mathsf{rch}_M)(z_{i+1}) && (\mathsf{rch}_M(s_{i+1}) = \mathsf{rch}_M(z_{i+1})) \\
&= (\pi \circ \mathsf{rch}_M)(z_i a_{i+1}) && (z_{i+1} = z_i a_{i+1}) \\
&= (\pi \circ \delta_M)(\mathsf{rch}_M(z_i), a_{i+1}) && \text{(definition of } \mathsf{rch}_M) \\
&= (\pi \circ \delta_M)(\mathsf{rch}_M(s_i), a_{i+1}) && (\mathsf{rch}_M(s_i) = \mathsf{rch}_M(z_i)) \\
&= \xi_\delta(s_i, a_{i+1}) && \text{(definition of } \xi_\delta) \\
&= \mathsf{cons}_\delta(e(s_i), a_{i+1}) && \text{(definition of } \mathsf{cons}_\delta) \\
&= \mathsf{cons}_\delta(\mathsf{rch}_H(z_i), a_{i+1}) && \text{(induction hypothesis)} \\
&= (m \circ \delta_H)(\mathsf{rch}_H(z_i), a_{i+1}) && \text{(definition of } \delta_H) \\
&= (m \circ \mathsf{rch}_H)(z_i a_{i+1}) && \text{(definition of } \mathsf{rch}_H) \\
&= (m \circ \mathsf{rch}_H)(z_{i+1}) && (z_{i+1} = z_i a_{i+1}).
\end{aligned}$$

This concludes the proof of (5).

In particular, then, $(m \circ e)(s_n) = \xi(s_n) = (m \circ \mathsf{rch}_H)(z)$. Because m is injective, we have $e(s_n) = \mathsf{rch}_H(z)$. Therefore,

$$\begin{aligned}
\mathcal{L}_H(z) &= (\mathsf{out}_H \circ \mathsf{rch}_H)(z) && \text{(definition of } \mathcal{L}_H) \\
&= (\mathsf{out}_H \circ e)(s_n) && (e(s_n) = \mathsf{rch}_H(z)) \\
&= \xi_{\mathsf{out}}(s_n) && \text{(definition of } \mathsf{out}_H) \\
&= (\mathsf{out}_M \circ \mathsf{rch}_M)(s_n) && \text{(definition of } \xi_{\mathsf{out}}) \\
&= (\mathsf{out}_M \circ \mathsf{rch}_M)(z) && (\mathsf{rch}_M(s_n) = \mathsf{rch}_M(z)) \\
&= \mathcal{L}_M(z) && \text{(definition of } \mathcal{L}_M).
\end{aligned}$$

◀

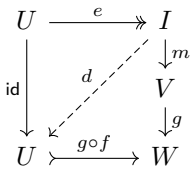
For the testing theorem, recall the wrappers from Section 6.

▶ **Theorem 20.** *Given the above wrappers, suppose $\sigma_U \in \mathcal{E}$, $\pi_U \in \mathcal{M}$, and either $\sigma_V \in \mathcal{E}$ and V is observable or $\pi_V \in \mathcal{M}$ and V is reachable. Then U is isomorphic to V if and only if all the equalities (4) hold.*

Proof. Assume first that U is isomorphic to V as witnessed by an isomorphism $\phi\colon U \to V$. By initiality, $\phi \circ \mathsf{rch}_U = \mathsf{rch}_V$; by finality, $\mathsf{obs}_V \circ \phi = \mathsf{obs}_U$. Then $\mathsf{obs}_U \circ \mathsf{rch}_U = \mathsf{obs}_V \circ \phi \circ \mathsf{rch}_U = \mathsf{obs}_V \circ \mathsf{rch}_V$. From this equality the conclusions (4) follow using Proposition 4. Now assume that the equalities (4) hold. From these equations we know that w_V must be init-closed, δ-closed, δ-consistent, and out-consistent, since w_U, satisfying $\sigma_U \in \mathcal{E}$ and $\pi_U \in \mathcal{M}$, has these properties by Lemma 11. Note that (4) also implies that the hypothesis automata of w_U and w_V coincide. Moreover, $\sigma_U \in \mathcal{E}$ and $\pi_U \in \mathcal{M}$ imply that U is minimal because of their definition and Proposition A.1 (and its dual). Using this, in combination with the assumptions for V, we can now apply Theorem 16 and conclude that U and V are isomorphic. ◀

▶ **Proposition A.1.** *Given an $(\mathcal{E}, \mathcal{M})$ factorization system on a category $\mathbf{C}$, if $g \circ f \in \mathcal{M}$, then $f \in \mathcal{M}$.*

Proof. Factorize $f = m \circ e$, with $e \in \mathcal{E}$ and $m \in \mathcal{M}$, and consider the unique diagonal d obtained from the commutative square below.

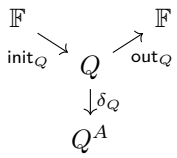


We see that $d \circ e = \mathsf{id}$. Then $e \circ d \circ e = e$, and therefore $e \circ d = \mathsf{id}$ because e is epic. Thus, e is an isomorphism. Since $\mathcal{M}$ is closed under composition with isomorphisms, we conclude that $f \in \mathcal{M}$. ◀

B Linear Weighted Automata

So far we have only considered DFAs as examples of automata. In this appendix, we exploit the categorical nature of CALF by changing the base category in order to study linear weighted automata.

Let $\mathbf{C}$ be the category $\mathbf{Vect}^{\mathbf{op}}$, the opposite of the category of vector spaces and linear maps over a field $\mathbb{F}$. We need the opposite category because the dynamics of our automata will be coalgebras rather than the algebras that are found in Definition 14. We interpret everything in $\mathbf{Vect}$ rather than $\mathbf{Vect}^{\mathbf{op}}$. The automata we consider are for $I = Y = \mathbb{F}$ and $F = (-)^A$, where the latter for a finite set A as an endofunctor on $\mathbf{Vect}$ is given for a vector space X by the set of functions X^A with a pointwise vector space structure. On linear maps $f \colon W \to X$ we have $f^A \colon W^A \to X^A$ given by $f^A(g)(a) = f(g(a))$. An automaton is now a vector space Q with linear maps as shown in the diagram. These automata are called *linear weighted automata* (LWAs).



Given a set U, let $V(U)$ be the free vector space generated by U, the elements of which are formal sums $\sum_{j \in J} v_j \times u_j$ for finite sets J, $\{v_j\}_{j \in J} \subseteq \mathbb{F}$, and $\{u_j\}_{j \in J} \subseteq U$. The operation $V(-)$ is a functor $\mathbf{Set} \to \mathbf{Vect}$ that assigns to a function $f \colon U \to W$ between sets U and W the linear map $V(f) \colon V(U) \to V(W)$ given by $V(f)(u) = f(u)$. In fact, the functor V is left adjoint to the forgetful functor $\mathbf{Vect} \to \mathbf{Set}$ that assigns to each vector space its underlying set. Given a vector space W and a function $f \colon U \to W$, the *linearization* of f is the linear map $\overline{f} \colon V(U) \to W$ given by $\overline{f}\left(\sum_{j \in J} v_j \times u_j\right) = \sum_{j \in J} v_j \times f(u_j)$. Conversely, a linear map with domain $V(U)$ is completely determined by its definition on the elements of U. The adjunction implies that this is a bijective correspondence between functions $U \to W$ and linear maps $V(U) \to W$. Note that $\mathbb{F}$ is isomorphic to $V(1)$, so the initial state map $\mathsf{init} \colon \mathbb{F} \to Q$ of an LWA Q is essentially an element of Q. This element is given by $\mathsf{init}(1)$. Moreover, any linear map is determined by its value on the basis vectors of its domain, which gives us a finite representation for LWAs with a finite-dimensional state space. The automaton in this representation is known as a *weighted automaton*.

The initial input system in this setting is essentially the same as for DAs in $\mathbf{Set}$, but enriched with a free vector space structure.

▶ **Proposition B.1.** *The vector space $V(A^*)$ with*

$$\mathsf{init}\colon \mathbb{F} \to V(A^*) \qquad \mathsf{init}(1) = \varepsilon \qquad \delta\colon V(A^*) \to V(A^*)^A \qquad \delta(u)(a) = ua$$

forms an initial input system.

Proof. Given an input system Q with initial state map $\mathsf{init}_Q\colon V(1) \to Q$ and dynamics $\delta_Q\colon Q \to Q^A$, define a linear map $\mathsf{rch}\colon V(A^*) \to Q$ by $\mathsf{rch}(\varepsilon) = \mathsf{init}_Q(1)$ and $\mathsf{rch}(ua) = \delta_Q(\mathsf{rch}(u))(a)$. Note that rch is an input system homomorphism. Any input system homomorphism $h\colon V(A^*) \to Q$ must satisfy $h(\varepsilon) = \mathsf{init}_Q(1)$ $h(ua) = \delta_Q(h(u))(a)$ and is therefore, by induction on the length of words, equal to rch. ◀

Hence, in this setting, reachability is defined in terms of formal sums of words, rather than just single words.

Observability maps are simply as they would be in **Set** [19, Lemma 7].

▶ **Proposition B.2.** *The vector space $\mathbb{F}^{A^*}$ with*

$$\mathsf{out}\colon \mathbb{F}^{A^*} \to \mathbb{F} \qquad \mathsf{out}(l) = l(\varepsilon) \qquad \delta\colon \mathbb{F}^{A^*} \to (\mathbb{F}^{A^*})^A \qquad \delta(l)(a)(u) = l(au)$$

forms a final output system.

The observability maps are just as for DAs, but generalized to an arbitrary output set. Thus, $\mathsf{obs}\colon Q \to \mathbb{F}^{A^*}$ is given by $\mathsf{obs}(q)(\varepsilon) = \mathsf{out}_Q(q)$ and $\mathsf{obs}(q)(au) = \mathsf{obs}(\delta_Q(q)(a))(u)$.

We use the (surjective linear maps, injective linear maps) factorization system in **Vect**. The details are similar to those in **Set**, but now the image of a linear map $f\colon U \to V$ is a subspace of V. To see that the unique diagonals are the same as in **Set**, note the following result.

▶ **Proposition B.3.** *For vector spaces U, W, and X, if $f\colon U \to W$ and $g\colon U \to X$ are linear maps and $h\colon X \to W$ is a function such that $h \circ f = g$ and f is surjective, then h is a linear map.*

Proof. Let J be a finite index set, $\{v_j\}_{j\in J} \subseteq \mathbb{F}$, and $\{w_j\}_{j\in J} \subseteq W$. Because f is surjective, we know that for each $j \in J$ there is a $u_j \in U$ such that $f(u_j) = w_j$. Then

$$\begin{aligned}
h\left(\sum_{j\in J} v_j \times w_j\right) &= h\left(\sum_{j\in J} v_j \times f(u_j)\right) && (f(u_j) = w_j)\\
&= h\left(f\left(\sum_{j\in J} v_j \times u_j\right)\right) && (\text{linearity of } f)\\
&= g\left(\sum_{j\in J} v_j \times u_j\right) && (h \circ f = g)\\
&= \sum_{j\in J} v_j \times g(u_j) && (\text{linearity of } g)\\
&= \sum_{j\in J} v_j \times h(f(u_j)) && (h \circ f = g)\\
&= \sum_{j\in J} v_j \times h(w_j) && (f(u_j) = w_j).
\end{aligned}$$

◀

A language in this setting is a linear map $\mathcal{L}\colon V(A^*) \to \mathbb{F}$. One obtains the minimal LWA for the language $\mathcal{L}$ by starting from the automaton $V(A^*)$ with output map $\mathcal{L}$ and taking the image of its observability map $t\colon V(A^*) \to \mathbb{F}^{A^*}$. Note that if we take the initial state map of $\mathbb{F}^{A^*}$ to be $t \circ \mathsf{init}_{V(A^*)}$, then t is an LWA homomorphism, and its image is an LWA because each category of automata has a factorization system inherited from the base category. It must be the minimal LWA because by initiality and finality the factorization of t must be $(\mathsf{rch}_M, \mathsf{obs}_M)$.

The dimension of a vector space U is denoted $\mathsf{dim}(U)$. The kernel of a linear map $f\colon U \to W$ is given by $\mathsf{ker}(f) = \{u \in U \mid f(u) = 0\}$. It is a standard result that if U is of finite dimension, then

$$\mathsf{dim}(U) = \mathsf{dim}(\mathsf{ker}(f)) + \mathsf{dim}(\mathsf{img}(f)). \tag{6}$$

Furthermore, if for two vector spaces U and W we have $U \subseteq W$, then $\mathsf{dim}(U) \leq \mathsf{dim}(W)$. If additionally $U \neq W$, then $\mathsf{dim}(U) < \mathsf{dim}(W)$.

For a linear map $f\colon U \to W$, if U is of finite dimension and $\mathsf{dim}(\mathsf{img}(f)) = \mathsf{dim}(U)$, then $\mathsf{dim}(\mathsf{ker}(f)) = 0$ by (6), implying f is injective. If $\mathsf{dim}(\mathsf{img}(f)) = \mathsf{dim}(W)$, then $\mathsf{img}(f) = W$, making f surjective.

Learning. We recover adaptations of $\mathtt{L}^\star$ and $\mathtt{ID}$ for LWAs as originally developed by Bergadano and Varricchio [8]. Jacobs and Silva [19] also instantiated their categorical reformulation of $\mathtt{L}^\star$ to this setting.

In active learning of LWAs, we assume there is a language $\mathcal{L}\colon V(A^*) \to \mathbb{F}$ such that the state space of the minimal LWA M accepting $\mathcal{L}$ is of finite dimension. Given a finite set $E \subseteq A^*$, the function $\mathsf{res}\colon \mathbb{F}^{A^*} \to \mathbb{F}^E$ given by $\mathsf{res}(L)(e) = L(e)$ is a linear map. To select elements of $V(A^*)$, we may take a finite subset $S \subseteq A^*$ and apply V to the inclusion $\mathsf{inc}\colon S \to A^*$ to obtain a linear map $V(\mathsf{inc})\colon V(S) \to V(A^*)$ and hence a wrapper

$$(\sigma, \pi) = (V(S) \xrightarrow{V(\mathsf{inc})} V(A^*) \xrightarrow{\mathsf{rch}} M, M \xrightarrow{\mathsf{obs}} \mathbb{F}^{A^*} \xrightarrow{\mathsf{res}} \mathbb{F}^E).$$

All approximated linear maps are as expected: $\xi\colon V(S) \to \mathbb{F}^E$ is given by $\xi(s)(e) = \mathcal{L}(se)$, $\xi_\delta\colon V(S) \to (\mathbb{F}^E)^A$ is given by $\xi_\delta(s)(a)(e) = \mathcal{L}(sae)$, $\xi_{\mathsf{out}}\colon V(S) \to \mathbb{F}$ is given by $\xi_{\mathsf{out}}(s) = \mathcal{L}(s)$, and $\xi_{\mathsf{init}}\colon \mathbb{F} \to \mathbb{F}^E$ is given by $\xi_{\mathsf{init}}(1)(e) = \mathcal{L}(e)$. Although these maps can be represented in the same kind of observation table that was used for learning a DFA (except that we have a different output set here), the notions of closedness and consistency are different. This is because we have to deal with the larger domains of the actual linear maps. The characterizations of these properties can still be derived from Proposition 10, as a result of Proposition B.3. It follows directly that δ-closedness holds if and only if for all $l \in V(S)$ and $a \in A$ there is an $l' \in V(S)$ such that $\xi(l') = \xi_\delta(l)(a)$. To simplify this, we have the following result.

▶ **Proposition B.4.** *If J is a finite set and $\{v_j\}_{j\in J} \subseteq \mathbb{F}$, $\{s_j\}_{j\in J} \subseteq S$, and $\{l_j\}_{j\in J} \subseteq V(S)$ are such that $\xi(l_j) = \xi_\delta(s_j)(a)$ for all $j \in J$, then*

$$\xi\left(\sum_{j\in J} v_j \times l_j\right) = \xi_\delta\left(\sum_{j\in J} v_j \times s_j\right)(a).$$

Proof. We have

$$\xi\left(\sum_{j\in J} v_j \times l_j\right) = \sum_{j\in J} v_j \times \xi(l_j) \qquad \text{(linearity of } \xi\text{)}$$

$$
\begin{aligned}
&= \sum_{j \in J} v_j \times \xi_\delta(s_j)(a) && (\xi(l_j) = \xi_\delta(s_j)(a)) \\
&= \left(\sum_{j \in J} v_j \times \xi_\delta(s_j) \right)(a) && \text{(pointwise vector space structure)} \\
&= \xi_\delta \left(\sum_{j \in J} v_j \times s_j \right)(a) && \text{(linearity of } \xi_\delta\text{).}
\end{aligned}
$$

◀

Thus, δ-closedness really says that for all $sa \in S \cdot A$ there must be an $l \in V(S)$ such that $\xi(l) = \xi_\delta(s)(a)$. As for δ-consistency, this property requires that for all $l_1, l_2 \in V(S)$ such that $\xi(l_1) = \xi(l_2)$ we must have $\xi_\delta(l_1)(a) = \xi_\delta(l_2)(a)$ for all $a \in A$. Using subtraction, this says that for all $l \in V(S)$ with $\xi(l) = \mathbf{0}$ we must have $\xi_\delta(l)(a) = \mathbf{0}$ for all $a \in A$, where $\mathbf{0} \colon E \to \mathbb{F}$ is given by $\mathbf{0}(e) = 0$ for all $e \in E$.

Analogously, init-closedness requires there to be an $l \in V(S)$ such that $\xi(l) = \xi_{\mathsf{init}}(1)$ while out-consistency states that for all $l \in V(S)$ such that $\xi(l) = \mathbf{0}$ we must have $\mathcal{L}(l) = 0$.

Closedness can be determined simply by solving systems of linear equations. Consistency is less trivial. Define the *transpose* of a table for the language $\mathcal{L}$ given by $S, E \subseteq A^*$ as the table with row labels $\mathsf{rev}(E)$ and column labels $\mathsf{rev}(S)$ for the language $\mathsf{rev}(\mathcal{L})$, where rev reverses all words in a language. We will show that consistency of a table can be ensured by ensuring closedness of the transposed table. This is essentially what Bergadano and Varricchio [8] do.

▶ **Proposition B.5.** *If the transpose of a table is δ-closed, then that table is δ-consistent.*

Proof. Suppose there is an $l \in V(S)$ such that $\xi(l) = \mathbf{0}$. For all $a \in A$ and $e \in E$ there are by closedness of the transposed table a finite set J, $\{v_j\}_{j \in J} \subseteq \mathbb{F}$, and $\{e_j\}_{j \in J} \subseteq E$ such that for every $s \in S$,

$$\xi_\delta(s)(a)(e) = \sum_{j \in J} v_j \times \xi(s)(e_j). \tag{7}$$

Let K be a finite set and $\{v'_k\}_{k \in K} \subseteq \mathbb{F}$ and $\{s_k\}_{k \in K} \subseteq S$ such that $l = \sum_{k \in K} v'_k \times s_k$. Then

$$
\begin{aligned}
\xi_\delta(l)(a)(e) &= \xi_\delta \left(\sum_{k \in K} v'_k \times s_k \right)(a)(e) && \text{(expanded form of } l\text{)} \\
&= \left(\sum_{k \in K} v'_k \times \xi_\delta(s_k) \right)(a)(e) && \text{(linearity of } \xi_\delta\text{)} \\
&= \sum_{k \in K} v'_k \times \xi_\delta(s_k)(a)(e) && \text{(pointwise vector space structure)} \\
&= \sum_{k \in K} v'_k \times \sum_{j \in J} v_j \times \xi(s_k)(e_j) && (7) \\
&= \sum_{j \in J} v_j \times \sum_{k \in K} v'_k \times \xi(s_k)(e_j) \\
&= \sum_{j \in J} v_j \times \left(\sum_{k \in K} v'_k \times \xi(s_k) \right)(e_j) && \text{(pointwise vector space structure)} \\
&= \sum_{j \in J} v_j \times \xi \left(\sum_{k \in K} v'_k \times s_k \right)(e_j) && \text{(linearity of } \xi\text{)}
\end{aligned}
$$

$$= \sum_{j \in J} v_j \times \xi(l)(e_j) \qquad \text{(expanded form of } l\text{)}$$
$$= \sum_{j \in J} v_j \times 0 \qquad (\xi(l) = \mathbf{0})$$
$$= 0. \qquad \blacktriangleleft$$

Analogously, one can show that init-closedness of the transposed table implies init-consistency of the original one.

If a closedness defect is found, we have a word $u \in A^*$ such that $(\mathsf{res} \circ \mathsf{obs}_M \circ \mathsf{rch}_M)(u)$ is not a linear combination of any row indexed by words from S. Because we have subtraction and scalar division, this also means that each of the rows indexed by S that is not a linear combination of other rows is also not a linear combination of other rows and $(\mathsf{res} \circ \mathsf{obs}_M \circ \mathsf{rch}_M)(u)$. Therefore, adding u to S increases the dimension of the hypothesis. The dimension of the hypothesis cannot exceed the dimension of the target M (see Appendix C), which is of finite dimension, so this process must terminate.

Note that the top part of a table in this setting can be seen as a matrix, and that transposing the table transposes this matrix. By fixing a closedness defect in the transpose of the table, the dimension of its hypothesis increases. This dimension is precisely the column rank of the original matrix. Since the column and row ranks of a matrix are equal, the dimension of the hypothesis of the original table must have increased.

The map $\sigma \colon V(S) \to M$ is surjective just if for each $m \in M$ there is an $l \in V(S)$ such that $\mathsf{rch}_M(l) = m$. Equivalently, the set $\{\mathsf{rch}_M(s) \mid s \in S\}$ needs to span M. Note that this means the set needs to include one of the (finite) bases of M. Thus, we have an adaptation of `ID` for LWAs that assumes to be given such a finite set S and enforces out-consistency and δ-consistency to obtain, by Theorem 16(1), a hypothesis isomorphic to M.

As for $\mathrm{L}^\star$, Proposition 17 carries over almost immediately. Due to the similarity, we omit the proof.

▶ **Proposition B.6.** *If for every prefix p of a word $z \in A^*$ there exists an $l \in V(S)$ such that $\mathsf{rch}_M(l) = \mathsf{rch}_M(p)$, then $\mathcal{L}_H(z) = \mathcal{L}_M(z)$.*

▶ **Corollary 7.** *If $z \in A^*$ is a counterexample, i.e., $\mathcal{L}_H(z) \neq \mathcal{L}_M(z)$, then adding all prefixes of z to S will increase the dimension of $\mathsf{img}(\mathsf{rch}_M \circ \mathsf{inc})$.*

Once the dimensions of $\mathsf{img}(\mathsf{rch}_M \circ \mathsf{inc})$ and M coincide, $\mathsf{rch}_M \circ \mathsf{inc}$ is surjective and we can apply Theorem 16(1). Thus, the number of required counterexamples is bounded by the dimension of M.

Reachability Analysis. Let Q be a finite-dimensional LWA and R its reachable part. Given a finite subset $S \subseteq R$, we have a wrapper $(V(S) \xrightarrow{\overline{\sigma}} R, R \xrightarrow{\pi} Q)$ in **Vect**, where σ and π are the inclusions. This wrapper is init-closed if there is an $l \in V(S)$ such that $\xi(l) = \mathsf{init}_Q(1)$, which can be satisfied by initializing $S = \{\varepsilon\}$. The wrapper is δ-closed if for all $s \in S$ and $a \in A$ there is an $l \in V(S)$ such that $\xi(l) = \delta_Q(\xi(s))(a)$. If for some sa this is not the case, we simply add it to S. As in learning, the dimension of the hypothesis increases by doing so, and therefore the process terminates. Using Theorem 16(2), the final hypothesis is isomorphic to R.

State Merging. Given a finite-dimensional LWA Q, we have a factorization of the observability map obs_Q as in Section 5—$Q \overset{h}{\twoheadrightarrow} O \overset{\mathsf{obs}}{\rightarrowtail} \mathbb{F}^{A^*}$—yielding an observable equivalent LWA O.

Fixing a finite set $E \subseteq A^*$, we have a wrapper $(Q \xrightarrow{h} O, O \xrightarrow{\mathsf{obs}} \mathbb{F}^{A^*} \xrightarrow{\mathsf{res}} \mathbb{F}^E)$. The linear map $\xi\colon Q \to \mathbb{F}^E$ gives the output of each state on the words in E. The wrapper is out-consistent if for all $q_1, q_2 \in Q$ such that $\xi(q_1) = \xi(q_2)$ we have $\mathsf{out}(q_1) = \mathsf{out}(q_2)$. If the last equation fails, we may add ε to E to distinguish $\xi(q_1)$ and $\xi(q_2)$. The wrapper is δ-consistent if for all $q_1, q_2 \in Q$ and $a \in A$ such that $\xi(q_1) = \xi(q_2)$ we have $\xi(\delta(q_1)(a)) = \xi(\delta(q_2)(a))$. If this last equation fails on some $e \in E$, we may add ae to E to distinguish $\xi(q_1)$ and $\xi(q_2)$. As in learning, this decreases the dimension of the kernel of ξ, which guarantees termination. Using Theorem 16(1), the final hypothesis is isomorphic to O.

Testing. Consider a known finite-dimensional LWA X and an unknown finite-dimensional LWA Z, both of which are minimal. Using the minimization algorithms (but with S for the reachability analysis a subset of A^*), we can find finite $S, E \subseteq A^*$ such that $\varepsilon \in S$ and the wrapper

$$w_X = (\sigma_X, \pi_X) = (V(S) \xrightarrow{V(\mathsf{inc})} V(A^*) \xrightarrow{\mathsf{rch}} X, X \xrightarrow{\mathsf{obs}} \mathbb{F}^{A^*} \xrightarrow{\mathsf{res}} \mathbb{F}^E)$$

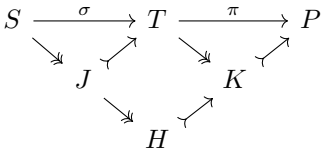
satisfies $\sigma_X \in \mathcal{E}$ and $\pi_X \in \mathcal{M}$. Define $w_Z = (\sigma_Z, \pi_Z)$ analogously. If at this point the equalities

$$\xi^{w_X} = \xi^{w_Z} \qquad \xi^{w_X}_{\mathsf{init}_X} = \xi^{w_Z}_{\mathsf{init}_Z} \qquad \xi^{w_X}_{\delta_X} = \xi^{w_Z}_{\delta_Z} \qquad \xi^{w_X}_{\mathsf{out}_X} = \xi^{w_Z}_{\mathsf{out}_Z}. \tag{8}$$

given by Theorem 20 do not hold, we can conclude that X and Z accept different languages. Assume these equalities do hold. By Corollary 13 this implies $\mathsf{dim}(\mathsf{img}(\xi^{w_Z})) = \mathsf{dim}(U)$. Assuming a given upperbound n on the dimension of Z, updating S to $S \cdot A^{\leq n - \mathsf{dim}(X)}$ yields $\sigma_Z \in \mathcal{E}$. Applying Theorem 20, we can find out whether X and Z are isomorphic by testing (8) for the updated wrappers. That is, we have an adaptation of the W-method for LWAs.

C The Hypothesis is Smaller than the Target

Given a wrapper $(S \xrightarrow{\sigma} T, T \xrightarrow{\pi} P)$ in an arbitrary category $\mathbf{C}$ with an $(\mathcal{E}, \mathcal{M})$ factorization system, the hypothesis can be defined using three successive factorizations, as shown below.



We now explain for the discussed concrete settings why the hypothesis is always "smaller" than the target, for the appropriate notion of size in each category.

In the Category of Sets. We know that if T is a finite set, then J, being a subset of T, is a finite set with $|J| \leq |T|$. Because there is a surjective function $J \to H$, we conclude that H is finite and $|H| \leq |J| \leq |T|$.

In the Category of Vector Spaces. Assume T is of finite dimension. Since J is a subspace of T, we have that J is a finite-dimensional vector space with $\mathsf{dim}(J) \leq \mathsf{dim}(T)$. Note that there exists a surjective linear map $x\colon J \to H$, so H is finite-dimensional and $\mathsf{dim}(\mathsf{img}(x)) = \mathsf{dim}(H)$. It then follows from (6) in Appendix B that $\mathsf{dim}(\mathsf{ker}(x)) = \mathsf{dim}(J) - \mathsf{dim}(H)$, so we must have $\mathsf{dim}(H) \leq \mathsf{dim}(J) \leq \mathsf{dim}(T)$.

Modal μ-Calculus with Atoms*

Bartek Klin[1] and Mateusz Łełyk[2]

1 **University of Warsaw, Warsaw, Poland**
2 **University of Warsaw, Warsaw, Poland**

Abstract

We introduce an extension of modal μ-calculus to sets with atoms and study its basic properties. Model checking is decidable on orbit-finite structures, and a correspondence to parity games holds. On the other hand, satisfiability becomes undecidable. We also show some limitations to the expressiveness of the calculus and argue that a naive way to remove these limitations results in a logic whose model checking is undecidable.

1998 ACM Subject Classification F.4.1 [Mathematical Logic] Temporal Logic, D.2.4 [Software/-Program Verification] Model Checking

Keywords and phrases modal μ-calculus, sets with atoms

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.30

1 Introduction

Modal μ-calculus [1, 4, 5, 29] is perhaps the best known formalism for describing properties of labeled transition systems or Kripke models. It combines a simple syntax with a mathematically elegant semantics and it is expressive enough to specify many interesting properties of systems. For example, the property "[in the current state] the predicate p holds, and there exists a transition path where it holds again some time in the future" is defined by a μ-calculus formula:

$$p \wedge \Diamond \mu X.(p \vee \Diamond X).$$

Other similar formalisms, such as the logic CTL* [17], can be encoded in the modal μ-calculus. However, decision problems such as model checking ("does a given formula hold in a given (finite) model?") or satisfiability ("does a given formula hold in some model?") are decidable.

Formulas of the μ-calculus are built over some fixed set of basic predicates, such as p above, whose semantics is provided in every model. In principle the set of such basic predicates may be infinite, but this generality is hardly useful: since a formula of the μ-calculus is a finite object, it may only refer to finitely many predicates.

In modeling systems, this finiteness may sometimes seem restrictive. Real systems routinely operate on data coming from potentially infinite domains, such as numbers or character strings. Basic predicates observed about a system may reasonably include ones like "a number n was input", denoted here p_n, for every number n. If one considers properties such as "there exists a transition path where the currently input number is input again some time in the future", one is in trouble writing finite formulas to define them. Were infinitary connectives allowed in the formalism, the formula

$$\bigvee_{n \in \mathbb{N}} (p_n \wedge \Diamond \mu X.(p_n \vee \Diamond X))$$

* This work is supported by Poland's National Science Centre grant 2012/07/B/ST6/01497.

would do the job; however, few good properties of the μ-calculus transport to a naively construed infinitary setting. In particular, an obvious obstruction to any kind of decidability results is that a general infinitary formula is not readily presented as input to an algorithm.

In this paper we introduce μ-calculus with atoms, where infinitary propositional connectives are allowed in a restricted form that includes formulas such as the one above. Roughly speaking, basic properties in formulas, and indeed whole Kripke models, are assumed to be built of *atoms* from a fixed infinite set. Connectives in μ-formulas are indexed by sets that are potentially infinite but are finitely definable in terms of atoms. This makes formulas finitely presentable.

Several basic properties of the standard μ-calculus hold in the atom-based setting. Syntax and semantics of the calculus is defined in a standard way. The model checking problem remains decidable, although this is not trivial, as formulas and models are now, strictly speaking, infinite. A correspondence between μ-formulas and parity games with atoms holds.

Some other properties of the classical calculus fail. The satisfiability problem becomes undecidable. The atomic μ-calculus also turns out to be less expressive than might be expected. In particular, the property "there exists a path where no basic predicate holds more than once", although decidable, is not definable. This means that, unlike in the classical setting, an atomic extension of CTL* (where explicit quantification over paths makes such properties easy to define) is not a fragment of the atomic μ-calculus. As it turns out, for atomic CTL* even the model-checking problem is undecidable.

Our approach is a part of a wider programme of extending various computational models to sets with atoms, also known as nominal sets [34]. For example, in [2] the classical notion of finite automaton was reinterpreted in the universe of sets with atoms, with the result related to register automata [21, 31], an established model of automata over infinite alphabets.

Temporal logics over structures extended with data from infinite domains, and their connections to various types of automata, have been extensively studied in the literature. For example, the linear time logic LTL has been extended with a *freeze quantifier* [15, 16, 20, 35] which, for structures where every position is associated with a single data item, can store the current item for future reference. This can serve as a mechanism for detecting repeated data values (see also [13, 14]). Another known idea is to extend temporal logics with local constraints over data from a fixed infinite domain, see e.g. [7, 12]. In [19, 25], alternating register automata on data words and trees were studied, closely connected to μ-calculi.

In all these works, the main goal is to study the decidability border for satisfiability (or nonemptiness, in the case of automata). To that end, the authors impose various, sometimes complex restrictions on their logics or automata, and inevitably limit their expressiveness to some extent. In contrast to that, our aim is to lift the classical modal μ-calculus to data-equipped structures in the most syntactically economic way, and to achieve an expressive formalism. As a price for this, the satisfiability problem quickly becomes undecidable. However, we believe that this does not disqualify the atomic μ-calculus as a practical formalism: most applications of temporal logics in system verification rely only on solving the model checking problem, and that remains decidable here on a wide class of structures.

It would be easy to give up even the decidability of model checking. This was done in [33], in a setting very similar to ours, by extending a basic multimodal logic with infinitary boolean connectives subject only to a finite support condition. The resulting logic has few good properties except its huge expressive power (indeed, it can immediately encode our atomic μ-calculus, and much more): its set of formulas is not even countable. Instead, our boolean connectives are subject to a more restrictive condition of orbit finiteness, an idea first used in [3] in the context of first-order logic with atoms.

Another branch of related work is centered around algebras of name-passing processes such as the π-calculus, and variants of the μ-calculus aimed at specific transition systems induced by those algebras. This line of work was started in [10], where a version of the μ-calculus for model checking properties of π-calculus processes was proposed, with a sound-and-complete proof system. Other efforts in this direction, resulting in logics fine-tuned to specific infinite models, include [18, 30], and a more abstract calculus was proposed in [11]. Finally, [23, 24] introduced a μ-calculus extended with predicates over arbitrary infinite data domains, with no hope for any general decidability results and with a focus on pragmatic usability.

The structure of this paper is as follows. In Sect. 2 we briefly recall the modal μ-calculus and related logics in the classical setting. In Sect. 3 we recall the basics of sets with atoms, including the notions of finite support and orbit finiteness. In Sect. 4 we introduce the syntax and semantics of the atomic μ-calculus. In Sect. 5 we prove that model-checking is decidable, and study a correspondence with parity games. In Sect. 6 we prove various undecidability results, and in Sect. 7 we study the undefinable path property mentioned above. A list of future work directions is in Sect. 8.

2 μ-calculus and related logics

To fix the notation and terminology, we begin by recalling basic definitions and properties of the μ-calculus in the classical setting. For a more detailed exposition see e.g. [1, 4–6, 36, 38].

▶ **Definition 2.1** (Syntax). Let $\mathbb{P}$ be an infinite set of *basic propositions* and $\mathbb{X}$ an infinite set of *variables*. The set $\mathcal{L}_\mu$ of μ-calculus formulas is generated by the grammar:

$$\phi ::= p \mid X \mid \phi \vee \phi \mid \neg\phi \mid \Diamond\phi \mid \mu X.\phi$$

where p ranges over $\mathbb{P}$ and X over $\mathbb{X}$. We only allow formulas $\mu X.\phi$ where X occurs only positively in ϕ (i.e. under an even number of negations).

We put $\top = p \vee \neg p$, $\phi \wedge \psi := \neg(\neg\phi \vee \neg\psi)$, $\Box\phi := \neg\Diamond\neg\phi$, $\nu X.\phi := \neg\mu X.\neg\phi[X := \neg X]$.

▶ **Definition 2.2** (Kripke model). A *Kripke model* is a triple $\langle K, R, W\rangle$ such that

1. $\langle K, R\rangle$ is a directed graph, i.e., R is a binary relation on the set K of *states*,
2. W is a function from $\mathbb{P}$ to $\mathcal{P}(K)$.

If $\mathcal{K} = \langle K, R, W\rangle$ is a Kripke model and $k \in K$ then $\langle \mathcal{K}, k\rangle$ is called a *pointed Kripke model*.

▶ **Definition 2.3** (Semantics). Formulas of μ-calculus are interpreted in the context of a Kripke model $\mathcal{K} = \langle K, R, W\rangle$ and an environment, i.e., a partial function $\rho : \mathbb{X} \rightharpoonup \mathcal{P}(K)$. For any formula ϕ, assuming ρ is defined on all free variables in ϕ, the interpretation $[\![\phi]\!]_\rho \subseteq K$ is defined by induction:

- $[\![p]\!]_\rho = W(p)$,
- $[\![X]\!]_\rho = \rho(X)$,
- $[\![\neg\phi]\!]_\rho = K \setminus [\![\phi]\!]_\rho$,
- $[\![\phi \vee \psi]\!]_\rho = [\![\phi]\!]_\rho \cup [\![\psi]\!]_\rho$,
- $[\![\Diamond\phi]\!]_\rho = \{k \in K \mid \exists s \in [\![\phi]\!]_\rho.\ \langle k, s\rangle \in R\}$,
- $[\![\mu X.\phi]\!]_\rho = \bigcap\{L \subseteq K \mid [\![\phi]\!]_{\rho[X\mapsto L]} \subseteq L\}$,

where $\rho[X \mapsto L]$ denotes the function that maps X to L and acts as ρ on all other arguments.

We write $[\![\phi]\!]$ (or $[\![\phi]\!]_\mathcal{K}$ if $\mathcal{K}$ is less clear from context) instead of $[\![\phi]\!]_\rho$ if ρ is empty. We say that ϕ *holds* in a state $k \in K$ if $k \in [\![\phi]\!]$ and denote it $k \models \phi$.

▶ **Example 2.4.** The formula $\mu X.(p \vee \Diamond X)$ holds for pointed Kripke structures $(\mathcal{K}, k)$ such that from k there is a finite path to a state where p holds. The formula $\nu X.(\Box X \wedge \mu Y.(p \vee \Box Y))$ holds in those pointed Kripke structures where on every path p holds infinitely often.

For applications in system verification, the following two problems are considered:

Model checking: given a finite pointed Kripke model $\langle \mathcal{K}, k \rangle$ and a formula $\phi \in \mathcal{L}_\mu$, decide if $k \models \phi$.

Satisfiability: given a formula $\phi \in \mathcal{L}_\mu$, decide if there exists a pointed Kripke model $\langle \mathcal{K}, k \rangle$ s.t. $k \models \phi$.

It is very easy to see that model checking is decidable: in a finite Kripke model, one can compute the semantics of an $\mathcal{L}_\mu$-formula inductively, calculating least fixpoints using approximants. A more efficient procedure can be derived from a correspondence of μ-calculus with parity games (see e.g. [6, 38]).

Some well-known logics used in system verification can be translated into fragments of the μ-calculus. We give two important examples:

▶ **Definition 2.5** (CTL*)**.** In the logic CTL* we distinguish state formulas Φ and path formulas ϕ, formed according to the following grammar:

$$\Phi ::= p \mid \Phi \vee \Phi \mid \neg\Phi \mid \exists\phi \qquad\qquad \phi ::= \Phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathsf{U} \phi \mid \mathsf{X}\phi$$

where p comes from some fixed set of propositional variables.

Standard notational conventions are $\forall\phi = \neg\exists\neg\phi$, $\phi \mathsf{R} \psi = \neg(\neg\phi \mathsf{U} \neg\psi)$, $\mathsf{F}\phi = \top \mathsf{U} \phi$, $\mathsf{G}\phi = \neg\mathsf{F}\neg\phi$.

CTL* formulas are interpreted in Kripke models $\mathcal{K}$: state formulas are interpreted over states, and path formulas over paths. In the following definition of a satisfaction relation $\models$, for a path $\pi = \langle k_0, k_1, k_2, \ldots \rangle$, by $\pi[n..]$ we denote the subpath starting at k_n.

- $k \models p \iff k \in W(p)$.
- $k \models \exists\phi \iff$ for some path π starting at $k, \pi \models \phi$.
- $\pi \models \Phi \iff \pi[0] \models \Phi$.
- $\pi \models \mathsf{X}\phi \iff \pi[1..] \models \phi$
- $\pi \models \phi \mathsf{U} \psi \iff$ there exists $j \geq 0$ s.t. $\pi[j..] \models \psi$ and for all $i < j$, $\pi[i..] \models \phi$.

Boolean connectives are interpreted as expected.

The logic LTL is the fragment of CTL* where the symbol $\exists$ does not occur, with semantics inherited from CTL*. Usually LTL (where the distinction between state and path formulas disappears) is interpreted in models that are infinite paths. Slightly more generally (and more conveniently for our purposes), we will interpret them over pointed *deterministic* Kripke models, i.e., ones where for each $k \in K$ there is exactly one $s \in K$ such that $\langle k, s \rangle \in R$. This makes little difference, since in such a model every state uniquely determines an infinite path.

3 Sets with atoms

We now recall the basic notions and results concerning sets with atoms, also known as nominal sets [34]. There are several essentially equivalent ways to introduce these; we follow the set-theoretic presentation of [22], culminating in the notion of orbit-finite sets [2, 34] and computable operations on them.

Fix a countably infinite set $\mathbb{A}$, whose elements we shall call *atoms*. A bijection on $\mathbb{A}$ will be called an atom automorphism, and the group of atom automorphisms is denoted $\mathrm{Aut}(\mathbb{A})$.

Loosely speaking, a set with atoms is a set that can have atoms, or other sets with atoms, as elements. Formally, the universe $\mathcal{U}^{\mathbb{A}}$ of sets with atoms is defined by a von Neumann-like hierachy, by transfinite induction on ordinal numbers α:

$$\mathcal{U}^{\mathbb{A}}_0 = \emptyset, \qquad \mathcal{U}^{\mathbb{A}}_{\alpha+1} = \mathcal{P}(\mathcal{U}^{\mathbb{A}}_\alpha) + \mathbb{A}, \qquad \mathcal{U}^{\mathbb{A}}_\beta = \bigcup_{\alpha<\beta} \mathcal{U}^{\mathbb{A}}_\alpha \quad \text{for } \beta \text{ a limit ordinal,}$$

where $+$ denotes disjoint union of sets.

We are interested in sets that depend only on a finite number of atoms, in the following sense. Atom automorphisms act on $\mathcal{U}^{\mathbb{A}}$ by consistently renaming all atoms in a given set. Formally, this is again defined by transfinite induction. This defines a group action:

$$\_ \cdot \_ : \mathcal{U}^{\mathbb{A}} \times \mathrm{Aut}(\mathbb{A}) \to \mathcal{U}^{\mathbb{A}}.$$

For a finite set $S \subset \mathbb{A}$, let $\mathrm{Aut}_S(\mathbb{A})$ be the group of those automorphisms of $\mathbb{A}$ that fix every element of S. We say that S *supports* a set x if $x \cdot \pi = x$ for every $\pi \in \mathrm{Aut}_S(\mathbb{A})$. A set is *equivariant* if it is supported by the empty set. If x has a finite support then it has the least finite support (see [34] for a proof), denoted $\mathrm{supp}(x)$.

Relations, functions etc. are sets in the standard sense, so the notions of support and equivariance applies to them as well. Unfolding the definitions, for equivariant sets X and Y, a relation $R \subseteq X \times Y$ is equivariant if $\langle x, y\rangle \in R$ implies $\langle x \cdot \pi, y \cdot \pi\rangle \in R$ and a function $f : X \to Y$ is equivariant if $f(x \cdot \pi) = f(x) \cdot \pi$, for every $\pi \in \mathrm{Aut}(\mathbb{A})$.

From now on, we shall only consider sets with atoms that are *hereditarily finitely supported*, i.e., ones that have a finite support, whose every element has some finite support and so on.

For any x with atoms, the *S-orbit* of x is the set $\{x \cdot \pi \mid \pi \in \mathrm{Aut}_S(\mathbb{A})\}$. For example, if S supports x then the S-orbit of x is the singleton $\{x\}$.

For any S, S-orbits form a partition of the universe $\mathcal{U}^{\mathbb{A}}$. Moreover, for any S-supported set X, the S-orbits of its elements form a partition of X. We call such X *S-orbit-finite* if it is a union of finitely many S-orbits. If $S \subseteq T$ are finite, S supports X and X is S-orbit-finite, then (T supports X and) X is also T-orbit-finite. Thanks to this observation, we may drop the qualifier and simply call X *orbit-finite*, meaning "S-orbit-finite for any/every S that supports X".

▶ **Example 3.1.**

- Any classical set (without atoms) is an equivariant set with atoms. Since its every element is also equivariant, it forms its own orbit. Therefore, a classical set is orbit-finite if and only if it is finite.
- An atom $a \in \mathbb{A}$ has no elements, and it is supported by $\{a\}$. Every finite set of atoms $S \subseteq \mathbb{A}$ is supported by S, and every element of it forms a singleton S-orbit. Its complement $\mathbb{A} \setminus S$ is also supported by S, and is a single S-orbit. A subset of $\mathbb{A}$ that is neither finite nor co-finite, is not finitely supported.
- The set $\mathbb{A}$ of atoms, the set $\binom{\mathbb{A}}{2}$ of two-element sets of atoms, and the set $\mathbb{A}^2$ of all ordered pairs of atoms, are equivariant sets. The first two have a single orbit each, and the last one is a union of two orbits:

  $$\mathbb{A}^2 = \{\langle a, a\rangle \mid a \in \mathbb{A}\} \cup \{\langle a, b\rangle \mid a \neq b \in \mathbb{A}\}.$$

  Similarly, $\mathbb{A}^n$ is orbit-finite for every $n \in \mathbb{N}$. The set $\mathbb{A}^*$ of finite sequences of atoms is hereditarily finitely supported, but not orbit-finite. The powerset $\mathcal{P}(\mathbb{A})$ is equivariant itself, but it contains elements that are not finitely supported, and therefore is not considered a legal set with atoms.

- There are four equivariant binary relations on $\mathbb{A}$: the empty relation, the equality relation, the inequality relation and the full relation. .
- There is no equivariant function from $\binom{\mathbb{A}}{2}$ to $\mathbb{A}$, but $\{\langle\{a,b\},a\rangle \mid a \neq b \in \mathbb{A}\}$ is a legal equivariant relation, and the function constant at an atom a is supported by $\{a\}$. The only equivariant function from $\mathbb{A}$ to $\mathbb{A}$ is identity. The only equivariant functions from $\mathbb{A}^2$ to $\mathbb{A}$ are the two projections, and the only equivariant function from $\mathbb{A}$ to $\mathbb{A}^2$ is the diagonal $a \mapsto \langle a,a\rangle$.

Orbit-finite sets, although usually infinite, can be presented by finite means and are therefore amenable to algorithmic manipulation. There are a few ways to do this. In [2] (with the idea going back to [8]), it was observed that every single-orbit equivariant set is in an equivariant bijection with a set of k-tuples of distinct atoms, suitably quotiented by a subgroup $G \leq \mathrm{Sym}(k)$ of the symmetric group on k elements. Thus such a set can be presented by a number k and the finite group G, and an orbit-finite set is a formal union of such presentations. A somewhat more readable and concise scheme was used in [26], where orbit-finite sets are presented by set-builder expressions of the form

$$\{e \mid v_1, \ldots, v_n \in \mathbb{A}, \phi\}$$

where e is again an expression, v_i are bound atom variables and ϕ is a first-order formula with equality. We refer to [26] for a precise formulation (and to [32] for a proof that all orbit-finite sets can be presented this way); suffice it to say that the expressions in Example 3.1 are of this form, and other similar expressions are allowed. The set defined by such an expression is supported by the atoms that appear freely in the expression.

It is not entirely trivial whether two representations define the same set or not. For example, the two-orbit equivariant set $\mathbb{A}^2$ from Example 3.1 can be represented in two different ways (with some light syntactic sugar added for representing ordered pairs):

$$\{\langle a,b\rangle \mid a,b \in \mathbb{A}, \top\} \qquad \text{or} \qquad \{\langle a,a\rangle \mid a \in \mathbb{A}, \top\} \cup \{\langle a,b\rangle \mid a,b \in \mathbb{A}, a \neq b\}.$$

However, set equality and other basic operations on orbit-finite sets are computable on their representations, including:

- checking whether one set is an element (or a subset) of another,
- union and intersection of sets, cartesian product, set difference,
- applying an orbit-finite function to an argument, composing functions or relations,
- finding the image of a subset along a relation,
- checking whether a finite set S supports a given set, calculating the least support of a set,
- partitioning a given set into S-orbits, calculating the S-orbit of a given element.

These basic operations have been implemented as components of atomic programming languages [27, 28].

4 μ-calculus with atoms

Syntactically, the μ-calculus with atoms (or *atomic μ-calculus*) is simply an extension of the classical formalism with orbit-finite propositional connectives.

From now on, fix a countably infinite set $\mathbb{X}$ of variables. (Until Remark 4 below it is convenient to think of it simply as a atom-free countable set as in the classical μ-calculus.)

▶ **Definition 4.1.** Let $\mathbb{P}$ be an equivariant set with atoms of basic propositions. The set $\mathcal{L}_\mu^{\mathbb{A}}$ of formulas of the atomic μ-calculus is generated by the following grammar:

$$\phi ::= p \mid X \mid \bigvee \Phi \mid \neg\phi \mid \Diamond\phi \mid \mu X.\phi$$

where p ranges over $\mathbb{P}$, X ranges over $\mathbb{X}$ and Φ ranges over *orbit-finite* sets of formulas. As usual we only allow $\mu X.\phi$ where X occurs only positively in ϕ.

The "orbit-finite set of formulas" above refers to a canonical action of $\mathrm{Aut}(\mathbb{A})$ on formulas, extending the action on $\mathbb{P}$ inductively. Note that, despite ostensibly infinite disjunctions, every formula in $\mathcal{L}_\mu^{\mathbb{A}}$ has a finite depth, since two formulas in the same orbit necessarily have the same depth. Thanks to this, no need arises for transfinite induction in reasoning about the syntax of $\mathcal{L}_\mu^{\mathbb{A}}$ formulas.

As expected, we put $\Box\phi$ as a shorthand for $\neg\Diamond\neg\phi$, $\nu X.\phi$ for $\neg\mu X.\neg\phi[X := \neg X]$ and $\bigwedge\Phi$ for $\neg\bigvee\{\neg\phi \mid \phi \in \Phi\}$. With these conventions, every formula can be written in the negation normal form, where negation occurs only in front of a basic proposition or a variable.

Often it is notationally convenient to view an orbit-finite set Φ as a family of formulas indexed by a simpler orbit-finite set. For example, we may write

$$\textstyle\bigvee_{a\in\mathbb{A}} \Diamond a \qquad \text{to mean} \qquad \bigvee\{\Diamond a \mid a \in \mathbb{A}\}.$$

▶ **Example 4.2.** Put $\mathbb{P} = \mathbb{A}$. For every $a \in \mathbb{A}$ let $\Phi_a = \{\neg b \mid b \in \mathbb{A} \setminus \{a\}\}$. Then Φ_a is supported by $\{a\}$ and orbit-finite, hence $\bigwedge\Phi_a := \bigwedge_{b\neq a} \neg b$ is a formula in $\mathcal{L}_\mu^{\mathbb{A}}$. The set

$$\Psi = \left\{\Diamond\left(a \wedge \textstyle\bigwedge_{b\neq a} \neg b\right) \mid a \in \mathbb{A}\right\}$$

is equivariant and orbit-finite, hence

$$\textstyle\bigwedge\Psi = \bigwedge_{a\in A} \Diamond\left(a \wedge \bigwedge_{b\neq a} \neg b\right)$$

is also a legal formula in $\mathcal{L}_\mu^{\mathbb{A}}$.

▶ Remark. In the classical μ-calculus, one often wants a formula to be *clean*, or *well-named*, meaning that every bound variable is bound only once. In the presence of infinitary connectives this may seem problematic. For example, in the formula

$$\textstyle\bigwedge_{a\in\mathbb{A}} (\mu X.a \vee \Diamond X)$$

the variable X occurs infinitely many times, and naively replacing each binding occurrence with a completely fresh variable would result in an orbit-infinite conjunction. An easy solution is to allow a nontrivial action of the group $\mathrm{Aut}(\mathbb{A})$ on the set of variables. One way to do this is to replace each (occurrence of) a bound variable with a formal occurrence of it as a subformula, i.e., a (finite) nested sequence of subformulas that contains that occurrence.

The following syntactic properties are easily proved by induction on the depth of formulas:

- Every formula ϕ is finitely supported.
- For every formula ϕ, the set of its subformulas is finitely supported and orbit-finite.
- The relation $\trianglelefteq \subseteq \mathcal{L}_\mu^{\mathbb{A}} \times \mathcal{L}_\mu^{\mathbb{A}}$ of being a subformula is equivariant, i.e. for every $\pi \in \mathrm{Aut}(\mathbb{A})$ and every $\phi, \psi \in \mathcal{L}_\mu^{\mathbb{A}}$ we have $\phi \trianglelefteq \psi$ if and only if $\phi \cdot \pi \trianglelefteq \psi \cdot \pi$.

Semantics of the atomic μ-calculus is a straightforward extension of the classical one: we simply require all sets and relations to be sets with atoms, and replace finite models with orbit-finite ones.

▶ **Definition 4.3.** An *atomic Kripke model* is a triple $\mathcal{K} = \langle K, R, W\rangle$ where
1. K is a set with atoms,
2. R is a finitely supported binary relation on K,
3. W is a finitely supported subset of $\mathbb{P} \times K$.

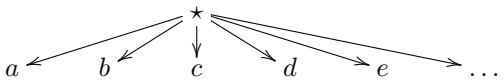
For every $p \in \mathbb{P}$ we denote $W(p) = \{k \in K \mid \langle p, k\rangle \in W\}$. Thus W can be equivalently seen as a finitely supported function from $\mathbb{P}$ to $\mathcal{P}_{\text{fs}}(K)$, the set of finitely supported subsets of K.

For any $k \in K$, the pair $\langle \mathcal{K}, k\rangle$ is called a *pointed Kripke model*. We shall say that an atomic Kripke model $\langle K, R, W\rangle$ is *orbit-finite* if the set K is so.

▶ **Example 4.4.** $\mathcal{K} = \langle K, R, W\rangle$, where:

$$K = \{\star\} \cup \mathbb{A},\ R = \{\langle \star, a\rangle \mid a \in \mathbb{A}\},\ W = \{\langle a, a\rangle \mid a \in \mathbb{A}\}$$

(where $\star$ is an equivariant element that forms a singleton orbit by itself) is an infinite but orbit-finite, equivariant atomic Kripke model which can be drawn as:



▶ **Definition 4.5.** For an atomic Kripke model $\mathcal{K} = \langle K, R, W\rangle$, the meaning of a formula $\phi \in \mathcal{L}_\mu^{\mathbb{A}}$ in a finitely supported variable environment $\rho : \mathbb{X} \rightharpoonup \mathcal{P}_{\text{fs}}(K)$ is defined exactly as in Definition 2.3, with the obvious modification:

$$\Big[\!\Big[\bigvee \Phi\Big]\!\Big]_\rho = \bigcup\{[\![\psi]\!]_\rho \mid \psi \in \Phi\}. \tag{1}$$

Following the philosophy of considering only finitely supported sets with atoms, the case of fixpoint formulas should also be modified to:

$$[\![\mu X.\phi]\!]_\rho = \bigcap\{L \subseteq K \mid L \text{ is finitely supported and } [\![\phi]\!]_{\rho[X\mapsto L]} \subseteq L\}. \tag{2}$$

The following easy lemma says that the meaning of every formula is a finitely supported set of states; in particular, it implies that if $\mathcal{K}$ is an equivariant model then for any formula ϕ without free variables, $[\![\phi]\!]_{\mathcal{K}}$ is supported by $\text{supp}(\phi)$.

▶ **Lemma 4.6.** *For every $\phi \in \mathcal{L}_\mu^{\mathbb{A}}$, an atomic Kripke model $\mathcal{K} = \langle K, R, W\rangle$ and a variable environment ρ, the set $[\![\phi]\!]_\rho \subseteq K$ is supported by $S = \text{supp}(\phi) \cup \text{supp}(\mathcal{K}) \cup \text{supp}(\rho) \subseteq \mathbb{A}$.*

Proof (sketch). By structural induction on ϕ. For example, consider the case of orbit-finite disjunction above, i.e., $\phi = \bigvee \Phi$. By the inductive assumption, the set $\{[\![\psi]\!]_\rho \mid \psi \in \Phi\}$ is supported by S. Since $\bigcup$ is an equivariant operation, the lemma follows.

The most interesting case is that of the fixpoint operator, $\phi = \mu X.\psi$. Recall the definition (2). The inclusion $[\![\psi]\!]_{\rho[X\mapsto L]} \subseteq L$, considered as a property of subsets $L \subseteq K$, is supported by S. Indeed, take any atom automorphism $\pi \in \text{Aut}_S(\mathbb{A})$. Since $\text{supp}(\phi) \subseteq S$, we have $\phi \cdot \pi = \phi$, hence $X \cdot \pi = X$ and $\psi \cdot \pi = \psi$. Since $\text{supp}(\rho) \subseteq S$, also $\rho \cdot \pi = \rho$. As a result, $[\![\psi]\!]_{\rho[X\mapsto L]} \subseteq L$ implies:

$$[\![\psi]\!]_{\rho[X\mapsto L\cdot\pi]} = [\![\psi \cdot \pi]\!]_{(\rho\cdot\pi)[X\cdot\pi\mapsto L\cdot\pi]} = ([\![\psi]\!]_{\rho[X\mapsto L]}) \cdot \pi \subseteq L \cdot \pi$$

where the last inclusion holds since the inclusion relation $\subseteq$ is equivariant.

Since the property of being finitely supported is equivariant, it follows that the family on the right of (2), hence its intersection, is supported by S. ◀

Atomic CTL* and atomic LTL are defined by analogy to atomic μ-calculus, extending Definition 2.5 with orbit-finite disjunctions, with semantics extended by analogy to (1) in Definition 4.5. With CTL* a design decision is to be made: in the semantic clause

- $k \models \exists\phi \iff$ for some path π starting at $k, \pi \models \phi$,

do we require the path π to be finitely supported or not? Note that even an equivariant Kripke model can contain infinite paths that are not finitely supported. We do not commit to a particular variant for now. Similar remarks apply to atomic LTL.

5 Basic properties

This section studies some basic results about the classical μ-calculus transported to the atomic setting. Proofs in this section are not difficult, but they illustrate standard techniques used to generalize properties of finite structures to orbit-finite ones.

5.1 Model-checking

▶ **Theorem 5.1.** *Model-checking problem for atomic μ-calculus over orbit-finite atomic Kripke models is decidable.*

Proof. Let us fix an orbit-finite atomic Kripke model $\mathcal{K} = \langle K, R, W\rangle$. We shall show that the meaning of any formula ϕ in $\mathcal{K}$ (under a variable environment ρ) can be computed from ϕ, by structural induction on ϕ and using basic operations listed at the end of Section 3.

The cases of basic propositions, variables, negation and the modality $\Diamond$ are straightforward. For the case of orbit-finite disjunction $\phi = \bigvee\Phi$, first calculate $S = \mathrm{supp}(\Phi) \cup \mathrm{supp}(\mathcal{K}) \cup \mathrm{supp}(\rho)$. Then partition Φ into S-orbits (there are finitely many of them), and select a system of representatives $\phi_1, \ldots, \phi_n$, one from each orbit. Using the inductive assumption, calculate $P_i = [\![\phi_i]\!]_\rho$ for each i. Then compute the S-orbit $\mathcal{O}_i$ of each P_i; each $\mathcal{O}_i$ is an S-supported family of subsets of K. The union of all $\bigcup\mathcal{O}_i$ is the desired set $[\![\phi]\!]_\rho$.

The most interesting case is computing $[\![\mu X.\phi]\!]_\rho$. This is done by approximating the least fixpoint by the following standard procedure:

(1) Put $L = \emptyset$,
(2) Extend ρ by mapping the variable X to L,
(3) Using the inductive assumption, calculate $[\![\phi]\!]_{\rho[X\mapsto L]}$ and put it as a new value of L,
(4) Repeat steps (2)-(3) until L stabilizes.

By the Knaster-Tarski theorem (see [1]) all we need to show is that this procedure terminates. Note that, by Lemma 4.6, each value assigned to L is a subset of K supported by $S = \mathrm{supp}(\phi) \cup \mathrm{supp}(\mathcal{K}) \cup \mathrm{supp}(\rho)$. In other words, L is the union of some selected S-orbits of K. Since K is orbit-finite, L can take on only finitely many values, therefore the above procedure terminates after finitely many steps. ◀

5.2 Parity games with atoms

The definition of atomic parity game is essentially as in the classical case (see e.g. [6, 38]):

▶ **Definition 5.2.** An *atomic parity game* $\mathcal{G}$ is a G quadruple $\langle V, V_\exists, R, \Omega\rangle$ such that

1. V is a set with atoms, $V_\exists$ is a finitely supported subset and we put $V_\forall = V \setminus V_\exists$;
2. $R \subseteq V^2$ is a finitely supported relation;
3. $\Omega : V \to \mathbb{N}$ is a bounded, finitely supported parity function.

The game is called *orbit-finite* if V is orbit-finite.

The notions of a match, partial match, strategy, positional strategy, winning strategy and determinacy are exactly as in the classical case.

Atomic parity games are obviously (forgetting about the action of atom automorphisms) parity games in the classical sense, so they are positionally determined. However, it might happen that in some atomic parity games no winning strategy is finitely supported.

▶ **Example 5.3.** Consider an atomic parity game where:

$$V = \binom{\mathbb{A}}{2} \cup \mathbb{A}, \quad V_\exists = \binom{\mathbb{A}}{2}, \quad R = \{\langle\{a,b\},a\rangle \mid a \neq b \in \mathbb{A}\} \cup \{\langle a,\{b,c\}\rangle \mid a,b,c \in \mathbb{A}, b \neq c\}$$
$$\Omega(v) = 0 \text{ for all } v \in V.$$

Since $\exists$ wins every infinite play and every state has a successor with respect to R, it is clear that every state is winning for $\exists$. However, no winning strategy for $\exists$ is finitely supported. Indeed, such a strategy would determine a finitely supported function from $\binom{\mathbb{A}}{2}$ to $\mathbb{A}$ such as $f(C) \in C$ for all $C \in \binom{\mathbb{A}}{2}$, and it is easy to see that no such function exists.

In spite of this, winning regions in orbit-finite parity games are computable. Indeed, every orbit-finite game can be effectively transformed into a finite game in the following way. For an orbit-finite parity game $\mathcal{G} = \langle V, V_\exists, R, \Omega\rangle$, let S be any finite set of atoms that supports $\mathcal{G}$. Let V/S, $V_\exists/S$ be the sets of S-orbits of V and $V_\exists$, respectively; let $[v]$ denote the S-orbit of $v \in V$. Obviously $V_\exists/S \subseteq V/S$. Define $R/S \subseteq V/S \times V/S$ and $\Omega/S : V/S \to \mathbb{N}$ by:

$$\langle [v],[w]\rangle \in R/S \quad \text{if} \quad \langle x,y\rangle \in R \text{ for some } x \in [v], y \in [w]$$
$$\Omega/S([v]) = n \quad \text{if} \quad \Omega(x) = n \text{ for some } x \in [v]$$

This is well defined since S supports both R and Ω. In particular, Ω/S is a function. We call $\mathcal{G}/S = \langle V/S, V_\exists/S, R/S, \Omega/S\rangle$ the *orbit game* of $\mathcal{G}$.

▶ **Lemma 5.4.** *The quotient function* Π *defined for every* $v \in V$ *by* $\Pi(v) = [v]$ *is a bisimulation between labeled Kripke models* $\langle V, R, \Omega\rangle$ *and* $\langle V/S, R/S, \Omega/S\rangle$.

Proof. First, if $\Pi(w) = [v]$, then $w \in [v]$ and consequently all the elements of $[v]$ have label $\Omega(w)$. So, by definition $\Omega/S([v]) = \Omega(w)$.

Now suppose $\Pi(w) = [v]$ and $\langle [v],[z]\rangle \in R/S$. It means that there are $x \in [v]$, $y \in [z]$ such that $\langle x,y\rangle \in R$. Since w and x are in the same S-orbit, pick a $\pi \in \mathrm{Aut}_S(\mathbb{A})$ such that $x \cdot \pi = w$. Since S supports R, we get $\langle w, y \cdot \pi\rangle \in R$. But $y \cdot \pi \in [z]$, so $\Pi(y \cdot \pi) = [z]$.

For the opposite direction, let $\Pi(w) = [v]$ and for some x, $\langle w,x\rangle \in R$. Then by the definition of R/S, $\langle [v],[x]\rangle \in R/S$ and obviously $\Pi(x) = [x]$. ◀

Moreover, for every $v \in V$, $v \in V_\exists$ if and only if $[v] \in V_\exists/S$. As a result, $\exists$ has a (positional) winning strategy from v in $\mathcal{G}$ if and only if she has a (positional) winning strategy from $[v]$ in $\mathcal{G}/S$. This implies that one can effectively decide whether a player has a winning strategy in an orbit-finite atomic parity game $\mathcal{G}$ by calculating first $S = \mathrm{supp}(\mathcal{G})$, then $\mathcal{G}/S$, and finally solving the problem in the finite parity game obtained, using standard methods.

A correspondence of the atomic μ-calculus with atomic parity games is proved essentially in the same way as in the classical case (see e.g. [6, 38]) using the notion of unfolding games. Together with the orbit game construction, this gives an alternative route to decidability of the model-checking problem for the atomic μ-calculus.

5.3 Failure of the orbit-finite model property

The classical modal μ-calculus enjoys the so-called finite model property: every satisfiable formula has a finite model. (In fact a stronger *small model property* holds, useful for complexity upper bounds.) There is no chance for this property to hold in the atomic setting, but since orbit-finite sets play the role of finite sets in the universe of sets with atoms, one might hope that an *orbit-finite model property* holds, i.e., that every satisfiable formula in $\mathcal{L}_\mu^{\mathbb{A}}$ has an orbit-finite model. However, even that weaker property fails, as we shall now prove.

Over a vocabulary of basic propositions that includes a proposition p_a for each atom a, consider the following three properties:

P1: every state reachable from the current state has at least one successor;

P2: in every state reachable from the current state some p_a holds;

P3: on every path starting in the current state, no p_a holds more than once.

All these are definable in the atomic μ-calculus. **P1** is simply $\nu X.(\Diamond\top \wedge \Box X)$, **P2** is $\nu X.\left(\bigvee_{a\in\mathbb{A}} p_a \wedge \Box X\right)$, and **P3** is

$$\neg(\mu X.(\psi \vee \Diamond X)) \qquad \text{where} \qquad \psi = \bigvee_{a\in\mathbb{A}} (p_a \wedge \Diamond \mu Y.(p_a \vee \Diamond Y)).$$

To build an atomic Kripke model for the conjunction of **P1**, **P2** and **P3**, put as states finite, nonempty sequences of distinct atoms. These form an equivariant set with atoms which, however, has infinitely many orbits (sequences of different lengths fall into separate orbits). For the transition relation put

$$R = \{\langle w, wa\rangle \mid a \notin w\},$$

that is, a sequence w can make a step to another valid sequence by appending a single atom a. The basic predicates are interpreted so that each p_a holds in exactly those sequences that end with the atom a. Properties **P1**, **P2** and **P3** hold in (every state of) this model.

However, the conjunction **P1**, **P2** and **P3** has no orbit-finite models. Indeed, assume that some state x_0 in such a model (with the transition relation denoted by R) satisfies all three properties. By **P1**, there exists an infinite path in the model:

$$x_0, x_1, x_2, x_3, x_4, \ldots \qquad \text{such that} \quad \langle x_i, x_{i+1}\rangle \in R.$$

By **P2**, each state on this path satisfies some predicate p_a, and by **P3** no such predicate is satisfied more than once.

Since the model is orbit-finite, there exists a global upper bound on the size of the least supports $\mathrm{supp}(x_i)$. This implies that there exists a number j and atoms $a \neq b$ such that:

- p_a holds in some x_i where $i < j$,
- p_b holds in some x_k where $j < k$, and
- $a, b \notin \mathrm{supp}(x_j)$.

Let $\pi \in \mathrm{Aut}(\mathbb{A})$ be the atom automorphism that swaps a and b and leaves all other atoms untouched; then $x_j \cdot \pi = x_j$, therefore $\langle x_{j-1}, x_j \cdot \pi\rangle \in R$. As a result:

$$x_0, x_1, x_2, \ldots, x_i, \ldots, x_{j-1}, x_j \cdot \pi, \ldots, x_k \cdot \pi, \ldots$$

is a legal path. But p_a holds both in x_i and in $x_k \cdot \pi$, so **P3** is violated on this path.

6 Undecidability results

▶ **Theorem 6.1.** *It is undecidable whether a given atomic LTL formula is satisfiable.*

Proof. See Appendix A. The proof follows the lines of the proof from [31] of the undecidability of the universality problem for register automata. ◀

▶ **Theorem 6.2.** *It is undecidable whether a given formula of the atomic μ-calculus is satisfiable.*

Proof. See Appendix B for details. Use a translation M from LTL into $\mathcal{L}_\mu^{\mathbb{A}}$ such that:

(i) In every word model, if a state satisfies ϕ then it satisfies $M(\phi)$,

(ii) In every Kripke model K, if a state x satisfies $M(\phi)$ then every path in K that starts from x, considered as a word model, satisfies ϕ.

Then apply Theorem 6.1. ◀

▶ **Theorem 6.3.** *The model checking problem for atomic CTL* is undecidable.*

Proof. Easy reduction from the satisfiability problem for atomic LTL; see Appendix C. ◀

7 Expressiveness limitations

In this section, consider Kripke models over a vocabulary of basic propositions that includes a proposition p_a for each atom a. For a state x in such a model, define $\mathrm{pred}(x) \subseteq \mathbb{A}$ to be the set of those atoms a for which p_a holds in x. Note that $\mathrm{pred}(x)$ ignores all propositions that are not of the form p_a.

Denote the property "there exists an infinite path where no p_a holds more than once", by #Path. Such properties of states in Kripke models have potentially significant practical importance. For example, one may imagine a system equipped with a token (e.g. password) generator where one needs to verify that, on every path where no token is generated more than once, the security of the system is never breached.

We shall show that although #Path is decidable, it is not definable in atomic μ-calculus. This is in contrast to the similar but definable property **P3** from Section 5.3.

▶ **Theorem 7.1.** #Path *is decidable on orbit-finite Kripke models.*

Proof. For simplicity, assume that a given orbit-finite Kripke model $\mathcal{K} = \langle K, R, W\rangle$ is equivariant; a generalization to finitely supported models is straightforward.

Notice that for every state $x \in K$, the set $\mathrm{pred}(x)$ is either finite (and contained in $\mathrm{supp}(x)$) or co-finite. Moreover, a single orbit of K only contains states of one of these two kinds. It is not difficult to decide the existence of a desired path where at least one state is of the second kind. Indeed, two such states cannot occur on the path at all, and even if exactly one of them occurs, almost all other states on the path must satisfy none of the predicates p_a. The existence of such a path from a given state x is straightforward to decide.

Once the existence of such paths is excluded, all (orbits of) states x with co-finite $\mathrm{pred}(x)$ may be safely deleted from the model. From now on, assume that $\mathrm{pred}(x) \subseteq \mathrm{supp}(x)$ for each $x \in K$.

Derived from $\mathcal{K}$, construct a new orbit-finite Kripke model $\hat{\mathcal{K}} = \langle \hat{K}, \hat{R}, \emptyset\rangle$, over the empty set of basic predicates, defined as follows:

$$\hat{K} = \{\langle x, S\rangle \mid x \in K,\ S \subseteq \mathrm{supp}(x) \setminus \mathrm{pred}(x)\}$$

$$\hat{R} = \{\langle\langle x, S\rangle, \langle y, T\rangle\rangle \mid\ \langle x, y\rangle \in R,\ (S \cup \mathrm{pred}(x)) \cap \mathrm{supp}(y) \subseteq T\}.$$

The intuition is that in a state $\langle x, S\rangle$, atoms in S are marked as having had occurred previously on a path, and are forbidden from occurring in the future. Note that this marking is restricted to atoms from the support of the current state x only.

In Appendix D we prove that a state $x \in K$ admits an infinite path where no p_a holds more than once, if and only if $(x, \emptyset) \in \hat{K}$ admits any infinite path. The theorem follows since the latter property is decidable (indeed, it is definable in the atomic μ-calculus, whose model-checking problem is decidable by Theorem 5.1). ◀

We shall now show that, in spite of its decidability and intuitive simplicity, #Path is not definable by a formula of the atomic μ-calculus. To this end, we introduce a hierarchy of bisimulations on atomic Kripke models.

Denote by $\mathbb{A}^{(\leq k)}$ the set of ordered tuples of pairwise distinct atoms of length at most k. Elements of such sets will be denoted with vector notation: $\vec{a}, \vec{b}$ etc. We shall write $x \sim y$ to say that x and y are in the same orbit.

▶ **Definition 7.2.** For a number $k \in \mathbb{N}$, a *k-bisimulation* on a Kripke model $\mathcal{K} = \langle K, R, W\rangle$ is a symmetric relation B on $K \times \mathbb{A}^{(\leq k)}$ such that, whenever $\langle x, \vec{a}\rangle B \langle y, \vec{b}\rangle$ then:

(i) $\langle \mathrm{pred}(x), \vec{a}\rangle \sim \langle \mathrm{pred}(y), \vec{b}\rangle$,
(ii) for every x' such that $\langle x, x'\rangle \in R$ there is a y' such that $\langle y, y'\rangle \in R$ and $\langle x', \vec{a}\rangle B \langle y', \vec{b}\rangle$,
(iii) for every $\vec{c} \in \mathbb{A}^{(\leq k)}$ there exists a $\vec{d} \in \mathbb{A}^{(\leq k)}$ such that $\langle \vec{a}, \vec{c}\rangle \sim \langle \vec{b}, \vec{d}\rangle$ and $\langle x, \vec{c}\rangle B \langle y, \vec{d}\rangle$.

Two states are called *k-bisimilar* if they are related by a k-bisimulation.

Some properties of k-bisimulations are straightforward to check. For example, by a standard argument, k-bisimilarity on a Kripke model is an equivalence relation. From condition (i) above it immediately follows that if $\langle x, \vec{a}\rangle$ and $\langle y, \vec{b}\rangle$ are k-bisimilar then $|\vec{a}| = |\vec{b}|$ and there exists a $\pi \in \mathrm{Aut}(\mathbb{A})$ such that $\vec{a} \cdot \pi = \vec{b}$. Furthermore, if shorter tuples $\vec{a}'$ and $\vec{b}'$ arise from $\vec{a}$ and $\vec{b}$ respectively by selecting the same subset of positions, then $\langle x, \vec{a}'\rangle$ and $\langle y, \vec{b}'\rangle$ are k-bisimilar as well. Finally, for $l < k$, the restriction of a k-bisimulation to the set $K \times \mathbb{A}^{(\leq l)}$ is an l-bisimulation.

The following result shows that k-bisimilar states cannot be distinguished by formulas from a certain fragment of the atomic μ-calculus. Call a formula $\phi \in \mathcal{L}_\mu^{\mathbb{A}}$ *globally k-supported* if every subformula of it (including ϕ itself) has a support of size at most k.

▶ **Theorem 7.3.** *For a globally k-supported formula ϕ, if*

- *$\langle x, \vec{a}\rangle$ and $\langle y, \vec{b}\rangle$ are k-bisimilar,*
- *$\pi \in \mathrm{Aut}(\mathbb{A})$ is such that $\vec{a} \cdot \pi = \vec{b}$,*
- *$\mathrm{supp}(\phi) \subseteq \vec{a}$, and $x \models \phi$*

then $y \models \phi \cdot \pi$.

▶ **Corollary 7.4.** *Assume that $\langle x, \epsilon\rangle$ and $\langle y, \epsilon\rangle$ are k-bisimilar. For every equivariant, globally k-supported formula ϕ, $x \models \phi$ if and only if $y \models \phi$.*

Proof. The Corollary is simply a special case of the Theorem, for $\vec{a} = \vec{b} = \epsilon$. To prove the Theorem, first apply a standard translation of the formula ϕ to infinitary modal logic, where no fixpoint operators are allowed, but boolean connectives of any arity are admitted. The translation is defined by induction. In the only interesting case, consider $\phi = \mu X.\psi$. For each ordinal α define a formula ϕ_α by:

$$\phi_0 = \bot, \qquad \phi_{\alpha+1} = \psi[X \mapsto \phi_\alpha], \qquad \phi_\beta = \bigvee_{\alpha<\beta} \phi_\alpha \text{ for a limiting ordinal } \beta.$$

It is then a standard result that $x \models \phi$ if and only if $x \models \bigvee_{\alpha<\kappa} \phi_\alpha$, where κ is the first cardinal larger than the Kripke model in question. Note that every ϕ_α is supported by

$\mathrm{supp}(\phi)$, and so is the entire infinite alternative. As a result, if an original formula ϕ is globally k-supported then the resulting infinitary formula is also globally k-supported.

From now on, assume that ϕ is an infinitary modal logic formula. The Theorem is proved by transfinite induction on the depth of ϕ, as follows:

- Assume $\phi = p_c$ for some basic predicate p_c. Since $x \models p_c$ (hence $c \in \mathrm{pred}(x)$) and $\langle x, \vec{a}\rangle$ and $\langle y, \vec{b}\rangle$ are k-bisimilar then, by condition (i) in Defn. 7.2, $c \cdot \pi \in \mathrm{pred}(y)$ and $y \models p_c \cdot \pi$.
- Assume $\phi = \Diamond\psi$. Then $x \models \phi$ means that $\langle x, x'\rangle \in R$ for some x' such that $x' \models \psi$. By condition (ii) in Defn. 7.2, there is some y' such that $\langle y, y'\rangle \in R$ and $\langle x', \vec{a}\rangle$ and $\langle y', \vec{b}\rangle$ are k-bisimilar. Since $\mathrm{supp}(\psi) = \mathrm{supp}(\phi) \subseteq \vec{a}$, by the inductive assumption we get $y' \models \psi \cdot \pi$. As a result, $y \models \Diamond(\psi \cdot \pi)$, and the latter formula equals $\phi \cdot \pi$.
- Assume $\phi = \neg\psi$, and towards a contradiction that $y \models \psi \cdot \pi$. Notice that $\mathrm{supp}(\psi \cdot \pi) \subseteq \vec{b}$. Applying the inductive assumption to $\psi \cdot \pi$, by the symmetry of k-bisimilarity we obtain $x \models \psi \cdot \pi\pi^{-1}$, and the latter formula is ψ, contradicting the assumption that $x \models \phi$.
- Assume $\phi = \bigvee \Phi$, where Φ is a set of formulas such that $\mathrm{supp}(\Phi) \subseteq \vec{a}$. Pick $\psi \in \Phi$ such that $x \models \psi$, and let $\vec{c}$ be the set $\mathrm{supp}(\psi)$, ordered in an arbitrary way. Note that $|\vec{c}| \leq k$ by the assumption that ϕ is globally k-supported. Pick a tuple $\vec{d}$ that exists by condition (iii) in Defn. 7.2. Since $\langle \vec{a}, \vec{c}\rangle \sim \langle \vec{b}, \vec{d}\rangle$, there exists a $\sigma \in \mathrm{Aut}(\mathbb{A})$ such that $\vec{a} \cdot \sigma = \vec{b}$ and $\vec{c} \cdot \sigma = \vec{d}$. Moreover, since $\langle x, \vec{c}\rangle$ and $\langle y, \vec{d}\rangle$ are k-bisimilar, we can apply the inductive assumption on ψ to obtain $y \models \psi \cdot \sigma$. This means that $y \models \phi \cdot \sigma$. However, since σ and π are equal on $\vec{a}$ and $\mathrm{supp}(\phi) \subseteq \vec{a}$, we get $\phi \cdot \sigma = \phi \cdot \pi$.
  Note that we did not assume that Φ is an orbit-finite set; the reasoning works for any set of formulas as long as it is supported by $\vec{a}$. ◀

Note that every formula of the atomic μ-calculus is globally k-supported for some number k. Therefore, by Cor. 7.4, to prove that #Path is not definable it is enough to construct, for every number k, an orbit-finite Kripke model $\mathcal{K} = \langle K, R, W\rangle$ over the language of basic predicate symbols $\{p_a \mid a \in A\}$, and two states $x, y \in K$ such that $\langle x, \epsilon\rangle$ and $\langle y, \epsilon\rangle$ are k-bisimilar and #Path fails for x but holds for y. To this end, choose any $S \subseteq \mathbb{A}$ s.t $|S| = 2k + 1$. Let $K = \mathbb{A}$ and define the transition relation R by:

$$\langle a, b\rangle \in R \iff (a \in S \iff b \in S).$$

In words, R forms a disjoint sum of a finite clique of size $2k + 1$ and an infinite clique. Interpret the basic predicates by:

$$a \models p_b \iff a = b.$$

Finally, pick some $x \in S$ and $y \notin S$. It is easy to see that #Path fails for x and holds for y.

It remains to be proved that $\langle x, \epsilon\rangle$ and $\langle y, \epsilon\rangle$ are k-bisimilar. To this end, consider a relation B on $K \times \mathbb{A}^{(\leq k)}$:

$$\langle u, \vec{a}\rangle B \langle v, \vec{b}\rangle \iff (|\vec{a}| = |\vec{b}|) \wedge (u \in S \iff v \notin S) \wedge \\ \wedge (\forall i.\ a_i \in S \iff b_i \notin S) \wedge (\forall i.\ a_i = u \iff b_i = v).$$

Obviously B is a symmetric relation and $\langle x, \epsilon\rangle B \langle y, \epsilon\rangle$. We shall show that B is a k-bisimulation by checking, for any $\langle u, \vec{a}\rangle B \langle v, \vec{b}\rangle$, the three conditions of Defn. 7.2 in turn:

(i) We have $\mathrm{pred}(u) = \{u\}$ and $\mathrm{pred}(v) = \{v\}$, and from the definition of B it immediately follows that $\langle \{u\}, \vec{a}\rangle \sim \langle \{v\}, \vec{b}\rangle$.

(ii) Consider any u' such that $\langle u, u'\rangle \in R$. If $u' = a_i$ for some i then put $v' = b_i$. On the other hand, if u' does not appear in $\vec{a}$ then take v' to be any atom that does not appear in $\vec{b}$ and such that $u' \in S \iff v' \notin S$. (For $u' \notin S$, this is possible since $|\vec{b}| \leq k < |S|$.) Either way, $\langle v, v'\rangle \in R$ and $\langle u', \vec{a}\rangle B \langle v', \vec{b}\rangle$ as required.

(iii) For any $\vec{c} \in \mathbb{A}^{(\leq k)}$ construct $\vec{d} \in \mathbb{A}^{(\leq k)}$ as follows:
- if $c_i = a_j$ for some (necessarily unique) j then put $d_i = b_j$,
- if $c_i = u$ then put $d_i = v$,
- otherwise put d_i to be any atom that does not occur in $\vec{b}$, is different from v and such that $c_i \in S$ iff $d_i \notin S$. At the same time, ensure that $d_i \neq d_j$ for $i \neq j$. This is possible since $|S| = 2k+1$, so there are at least k distinct atoms in S that do not occur in $\vec{b}$ and are different from v.

For $\vec{d}$ constructed this way, $\langle \vec{a}, \vec{c} \rangle \sim \langle \vec{b}, \vec{d} \rangle$ and $\langle u, \vec{c} \rangle B \langle v, \vec{d} \rangle$ as required.

The above Kripke model is not equivariant; its support is the chosen set S. However, a similar effect can be achieved in an equivariant model. Indeed, given a number k, it is enough to take the disjoint union, indexed by the family of all finite sets S of size $2k+1$, of the models as above. The resulting model is equivariant with $2k+2$ orbits of states, and the above reasoning holds for it without significant changes.

▶ Remark. The property #Path is easy to define in atomic CTL* with path formulas interpreted over arbitary paths, by a formula:

$$\exists \left(\bigwedge_{a \in \mathbb{A}} \mathsf{G}(p_a \to \mathsf{X}(\mathsf{G} \neg p_a)) \right).$$

This is, however, of little practical use due to Theorem 6.3. It also means that, contrary to the classical atom-less setting, atomic CTL* is not a fragment of the atomic μ-calculus.

8 Future work

We list some interesting aspects of the atomic μ-calculus that are best left for future work.

Complexity issues. For the decidability results we presented, in particular for the model checking problem over orbit-finite structures, one immediately asks about the complexity of the algorithms proposed. The answer depends on the way one measures the size of input structures. One obvious option is to consider the length of their representation with set-builder expressions and first-order formulas. With this view, most basic operations listed at the end of Section 3 become Pspace-hard, because the first-order theory of pure equality is Pspace-complete. As a result, the complexity of basic operations dwarfs the distinction between the two algorithmic approaches to model checking based on direct fixpoint computation and on parity games.

Another approach is to measure orbit-finite structures by the number of their orbits, and the (hereditary) size of their least support. Note that the number of orbits of a set can be exponentially bigger than the size of its logical representation; for example, the number of orbits of the set $\mathbb{A}^n$, whose representation has size linear in n, is equal to the n-th Bell number. In this view the difference between the two approaches to model checking becomes more prominent.

We defer precise complexity analyses until we have a better general understanding of various time and space complexity models on atomic structures.

Other atoms. As advocated in [2], much of the theory of sets with atoms can be generalized to other relational structures $\mathbb{A}$. For example, one can consider *ordered atoms* $\mathbb{A} = \langle \mathbb{Q}, < \rangle$, where sets such as the interval $\{b \in \mathbb{A} \mid 2 < b < 5\}$ become finitely supported. Other atom structures are also possible; see [2] for details.

If the first-order theory of $\mathbb{A}$ is decidable then most definitions and results in this paper generalize to other structures of atoms studied in [2]. An interesting exception is the

undefinability of #Path studied in Section 7. Our construction there does not work for ordered atoms; indeed, in the model $\mathcal{K}$ constructed after Corollary 7.4, the states $\langle x, \epsilon\rangle$ and $\langle y, \epsilon\rangle$ are not even 1-bisimilar.

The property #Path does not mention the order of atoms in any way, so it would be quite surprising if it turned out to be definable in the μ-calculus with ordered atoms. We leave this as an open problem.

Defining #Path. The fact that #Path is not definable in the μ-calculus with equality atoms is disappointing, since it looks like a property of potential practical importance in system verification. Since we know that #Path is decidable, it is desirable to extend atomic μ-calculus in some well-structured and syntactically economic way that would allow one to define such properties while preserving the decidability of model checking. The property of "global freshness" has been studied in the context of automata with atoms [37], and one may look for inspiration there. Our proof of Theorem 7.1 also suggests some promising options. We leave this for future work.

Acknowledgments. We are very grateful to A. Facchini for collaboration in initial stages of this work, to M. Bojańczyk, S. Lasota, S. Toruńczyk and B. Wcisło for valuable discussions, and to anonymous reviewers for their insightful comments.

References

1 A. Arnold and D. Niwiński. *Rudiments of μ-calculus*. Studies in logic and the foundations of mathematics. London, Amsterdam, 2001.

2 M. Bojańczyk, B. Klin, and S. Lasota. Automata theory in nominal sets. *Log. Meth. Comp. Sci.*, 10, 2014.

3 M. Bojańczyk and T. Place. Toward model theory with data values. In *Procs. ICALP 2012 Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 116–127, 2012.

4 J. Bradfield and C.Stirling. Modal mu-calculi. In *Handbook of Modal Logic*, volume 3, pages 721–756. Elsevier, 2007.

5 J. Bradfield and C. Stirling. Modal logics and mu-calculi: an introduction. In *Handbook of Process Algebra*, pages 293–330. North-Holland, 2001.

6 J. Bradfield and I. Walukiewicz. The mu-calculus and model-checking. In E. Clarke, T. Henzinger, and H. Veith, editors, *Handbook of Model Checking*. Springer-Verlag, 2015.

7 C. Carapelle and M. Lohrey. Temporal logics with local constraints (invited talk). In *Procs. CSL 2015*, volume 41 of *LIPIcs*, pages 2–13, 2015.

8 V. Ciancia and U. Montanari. Symmetries, local names and dynamic (de)-allocation of names. *Inf. Comput.*, 208(12):1349–1367, 2010.

9 S. Cranen, J. F. Groote, and M. Reniers. A linear translation from CTL* to the first-order modal μ-calculus. *Theoretical Computer Science*, 412(28):3129–3139, 2011.

10 Mads Dam. Model checking mobile processes. *Information and Computation*, 129(1):35–51, 1996.

11 Rocco De Nicola and Michele Loreti. Multiple-labelled transition systems for nominal calculi and their logics. *Mathematical Structures in Computer Science*, 18(01):107–143, 2008.

12 S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Inf. Comput.*, 205(3):380–415, 2007.

13 S. Demri, D. D'Souza, and R. Gascon. Temporal logics of repeating values. *J. Log. Comput.*, 22(5):1059–1096, 2012.

14 S. Demri, D. Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. *Logical Methods in Computer Science*, 12(3), 2016.

15 S. Demri and R. Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009.

16 S. Demri, R. Lazic, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. *Inf. Comput.*, 205(1):2–24, 2007.

17 E. A. Emerson and J. Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

18 Gian-Luigi Ferrari, Stefania Gnesi, Ugo Montanari, and Marco Pistore. A model-checking verification environment for mobile processes. *ACM Transactions on Software Engineering and Methodology*, 12(4):440–473, 2003.

19 D. Figueira. Alternating register automata on finite words and trees. *Logical Methods in Computer Science*, 8(1), 2012.

20 D. Figueira and L. Segoufin. Future-looking logics on data words and trees. In *Procs. MFCS 2009*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343, 2009.

21 N. Francez and M. Kaminski. Finite-memory automata. *TCS*, 134(2):329–363, 1994.

22 M. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.

23 Jan Friso Groote and Radu Mateescu. Verification of temporal properties of processes in a setting with data. In *Procs. AMAST'99*, pages 74–90, 1999.

24 Jan Friso Groote and Tim A. C. Willemse. Model-checking processes with data. *Science of Computer Programming*, 56(3):251–273, 2005.

25 M. Jurdzinski and R. Lazic. Alternating automata on data trees and xpath satisfiability. *ACM Trans. Comput. Log.*, 12(3):19:1–19:21, 2011.

26 B. Klin, E. Kopczyński, J. Ochremiak, and S. Toruńczyk. Locally finite constraint satisfaction problems. In *Procs. LICS 2015*, pages 475–486, 2015.

27 B. Klin and M. Szynwelski. SMT solving for functional programming over infinite structures. In *MFSP*, volume 207, pages 57–75, 2016. doi:10.4204/EPTCS.207.3.

28 E. Kopczyński and S. Toruńczyk. Lois: Syntax and semantics. In *Procs. of POPL 2017*, pages 586–598, 2017.

29 D. Kozen. Results on the propositional μ-calculus. *Theor. Comp. Sci.*, 27(3):333–354, 1983. doi:10.1016/0304-3975(82)90125-6.

30 Hui-Min Lin. Predicate μ-calculus for mobile ambients. *Journal of Computer Science and Technology*, 20(1):95–104, 2005.

31 F. Neven, T. Schwentick, and V. Vianu. Towards regular languages over infinite alphabets. In *MFCS*, pages 560–572, 2001.

32 J. Ochremiak. *Extended constraint satisfaction problems*. PhD thesis, University of Warsaw, 2016.

33 J. Parrow, J. Borgström, L.-H. Eriksson, R. Gutkovas, and T. Weber. Modal logics for nominal transition systems. In *Procs. CONCUR 2015*, volume 42 of *LIPIcs*, pages 198–211, 2015.

34 A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013.

35 L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Procs. CSL 2006*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57, 2006.

36 W. Thomas and T. Wilke. Automata, logics, and infinite games: A guide to current research. *Bulletin of Symbolic Logic*, 10(1):114–115, 2004.

37 N. Tzevelekos. Fresh-register automata. *SIGPLAN Not.*, 46(1):295–306, 2011.

38 Y. Venema. *Lectures on the modal μ-calculus*. ILLC, Univ. of Amsterdam, 2007.

A Proof of Theorem 6.1

Given a Turing machine $\mathcal{M}$ with a (finite) set of states Q, a (finite) working alphabet Γ (including a blank symbol $\flat$), initial and accepting states $q_{init}, q_{acc} \in Q$ and a transition relation δ, we shall build an atomic LTL formula that is satisfiable if and only if $\mathcal{M}$ accepts the empty word. Without loss of generality assume that $\mathcal{M}$, upon reaching an accepting configuration, enters an infinite loop in that configuration.

Consider the following language of basic predicate symbols:

- a special symbol \$,
- $\mathtt{atom}_a$ for each $a \in \mathbb{A}$,
- $\mathtt{tape}_\gamma$ for each $\gamma \in \Gamma$,
- $\mathtt{head}_q$ for each $q \in Q$.

Our formula shall be a conjunction of several properties. First, we enforce that every state in a model either satisfies \$ (and no other basic predicates) or satisfies exactly one predicate $\mathtt{atom}_a$, exactly one predicate $\mathtt{tape}_\gamma$ and at most one predicate $\mathtt{head}_q$. This is ensured by a conjunction of formulas such as:

$$\mathsf{G}\left(\$ \vee \bigvee_{a\in\mathbb{A}} \mathtt{atom}_a\right) \quad \text{and} \quad \bigwedge_{a\neq b\in\mathbb{A}} \mathsf{G}(\mathtt{atom}_a \to \neg\mathtt{atom}_b)$$

and so on. We also ensure that the initial state of the model satisfies \$.

Furthermore, we ensure that as far as predicates \$ and $\mathtt{atom}_a$ are concerned, the model can be presented as an infinite word:

$$\$w\$w\$w\cdots$$

where w is a finite sequence of predicate symbols of the form $\mathtt{atom_a}$ such that no single predicate appears in w more than once. This is achieved by a conjunction of formulas such as:

- $\bigwedge_{a,b\in\mathbb{A}} \mathsf{G}(\mathtt{atom}_a \wedge \mathsf{X}\mathtt{atom}_b \to \mathsf{G}(\mathtt{atom}_a \to \mathsf{X}\mathtt{atom}_b))$
- $\bigwedge_{a\in\mathbb{A}} \mathsf{G}(\mathtt{atom}_a \to (\neg\mathtt{atom}_a \mathsf{U} \$))$.

Portions of the model between two consecutive occurrences of \$ will store configurations of $\mathcal{M}$. To this end, we ensure that each portion contains exactly one state that satisfies some head predicate:

$$\mathsf{G}(\$ \to \mathsf{X}((\neg\psi \wedge \neg\$)\mathsf{U}(\psi \wedge \mathsf{X}(\neg\psi \mathsf{U} \$))))$$

where $\psi = \bigvee_{q\in Q} \mathtt{head}_q$.

We then ensure that every two consecutive configurations encode a legal step of $\mathcal{M}$. Predicates atom are useful for this, as they trace single tape cells in subsequent configurations. To ensure that the letters on the tape do not change unless the machine head is directly over them, we state for each $\gamma \in \Gamma$:

$$\bigwedge_{a\in\mathbb{A}} \mathsf{G}(\mathtt{atom}_a \wedge \mathtt{tape}_\gamma \wedge \neg\psi \to \mathsf{X}(\neg\mathtt{atom}_a \mathsf{U}(\mathtt{atom}_a \wedge \mathtt{tape}_\gamma)))$$

where ψ is as before. Furthermore, for every "head to the right" rule

$$\langle q, \gamma, q', \gamma', \Rightarrow\rangle \in \delta$$

we add a formula

$$\bigwedge_{a\in\mathbb{A}} \mathsf{G}(\mathtt{atom}_a \wedge \mathtt{head}_q \wedge \mathtt{tape}_\gamma \to \mathsf{X}(\neg\mathtt{atom}_a \mathsf{U}(\mathtt{atom}_a \wedge \mathtt{tape}_{\gamma'} \wedge \mathsf{X}\mathtt{head}_{q'})))$$

and similarly for "head to the left" transition rules.

Finally we ensure the correct form of the initial configuration and that an accepting state is reached, by adding formulas $\mathsf{X}\mathtt{head}_{q_{\mathtt{init}}}$, $\mathsf{X}(\mathtt{tape}_\flat \mathsf{U}\$)$ and $\mathsf{F}\mathtt{head}_{q_{acc}}$.

Models of the conjunction of all these formulas correspond to accepting runs of $\mathcal{M}$ on the empty input word. As a result, it is undecidable whether an LTL formula has a model.

B Proof of Theorem 6.2

For any LTL formula ϕ in the negation normal form (i.e., one where negations occur only in front of basic predicates, and where conjunction, disjunction, X, U and R modal operators are used), construct an atomic μ-calculus formula $M(\phi)$ by induction as follows:

$$M(\top)=\top, \quad M(\bot)=\bot, \quad M(p)=p, \quad M(\neg p)=\neg p, \quad M(\mathsf{X}\phi)=\Box M(\phi)$$
$$M(\phi\vee\psi)=M(\phi)\vee M(\psi), \quad M(\phi\wedge\psi)=M(\phi)\wedge M(\psi)$$
$$M(\phi\mathsf{U}\psi)=\mu Y.(M(\psi)\vee(M(\phi)\wedge\Box Y)), \quad M(\phi\mathsf{R}\psi)=\nu Y.(M(\psi)\wedge(M(\phi)\vee\Box Y))$$

This translation does not have all properties that may be desired (see e.g. [9]) but it is sufficient for our purposes:

(i) In every word model, if a state satisfies ϕ then it satisfies $M(\phi)$,

(ii) In every Kripke model K, if a state x satisfies $M(\phi)$ then every path in K that starts from x, considered as a word model, satisfies ϕ.

Note that the converse to the implication in (ii) does not hold in general (consider e.g. $\phi=\mathsf{X}p\vee\mathsf{X}q$ and its translation $M(\phi)=\Box p\vee\Box q$). Both properties (i) and (ii) are proved by induction, for example for (ii):

- Assume $x\models M(\phi\vee\psi)$. Without loss of generality, assume $x\models M(\phi)$. By the inductive assumption, every path starting from x satisfies ϕ, so it satisfies $\phi\vee\psi$.
- Assume $x\models M(\phi\mathsf{U}\psi)$. By definition of M, on every path π starting from x the formula $M(\phi)$ holds in every state y until at some point $M(\psi)$ holds. Now, for each state y the path π has a sub-path that starts at y, and by the inductive assumption ϕ holds for all these subpaths until at some point ψ holds. As a result, $\phi\mathsf{U}\psi$ holds for the path π.

Properties (i) and (ii) immediately imply that $M(\phi)$ is satisfiable if and only if ϕ is. By Theorem 6.1, satisfiability of atomic μ-calculus formulas is undecidable.

C Proof of Theorem 6.3

We reduce the satisfiability problem for atomic LTL, with some insight into the proof of Theorem 6.1. Given any Turing machine $\mathcal{M}$ as considered there, consider a Kripke model with the set of states:

$$(\mathbb{A}\times\Gamma\times(Q\cup\{\mathtt{nohead}\}))\cup\{\$\}$$

with basic predicates defined in the obvious way, and with transitions going both ways between every two states. This model is orbit-finite and equivariant.

Now, for the atomic LTL formula ϕ obtained from $\mathcal{M}$ as in the proof of Theorem 6.1, the formula $\exists\phi$ holds in this model in the state \$ if and only if ϕ is satisfiable.

This works regardless of whether we interpret CTL* path formulas over arbitrary paths or over finitely supported ones, because the formula ϕ in the proof of Theorem 6.1 forces its model to be finitely supported.

D Details of the proof of Theorem 7.1

Note that $\hat{\mathcal{K}}$ is indeed equivariant and orbit-finite: every orbit of K gives rise to at most 2^k orbits in $\hat{K}$, where k is the size of the least support of any (equivalently, every) element in the orbit. Moreover, (a representation of) $\hat{\mathcal{K}}$ is computable from $\mathcal{K}$: for each orbit in K one can enumerate all corresponding orbits in $\hat{K}$, and orbits of transitions in $\hat{R}$ are also easy to enumerate.

We prove that a state $x \in K$ admits an infinite path where no p_a holds more than once, if and only if $(x, \emptyset) \in \hat{K}$ admits any infinite path.

For the left-to-right implication, assume an infinite path in $\mathcal{K}$, i.e., a sequence $x = x_0, x_1, x_2, x_3 \ldots$, such that $\langle x_i, x_{i+1}\rangle \in R$ for each $i \in \mathbb{N}$, and $\mathrm{pred}(x_i) \cap \mathrm{pred}(x_j) = \emptyset$ for each $i \neq j \in \mathbb{N}$. Define

$$y_i = \langle x_i, S_i\rangle, \quad \text{where} \quad S_i = \mathrm{supp}(x_i) \cap \bigcup_{j=0}^{i-1} \mathrm{pred}(x_j).$$

In particular, $y_0 = \langle x, \emptyset\rangle$. Then

$$\langle\langle x_i, S_i\rangle, \langle x_{i+1}, S_{i+1}\rangle\rangle \in \hat{R} \tag{3}$$

for each $i \in \mathbb{N}$. Indeed, calculate

$$\begin{aligned}(S_i \cup \mathrm{pred}(x_i)) \cap \mathrm{supp}(x_{i+1}) &= \left(\left(\mathrm{supp}(x_i) \cap \bigcup_{j=0}^{i-1} \mathrm{pred}(x_j)\right) \cup \mathrm{pred}(x_i)\right) \cap \mathrm{supp}(x_{i+1}) \\ &\subseteq \left(\bigcup_{j=0}^{i} \mathrm{pred}(x_j)\right) \cap \mathrm{supp}(x_{i+1}) = S_{i+1}.\end{aligned}$$

As a result, the pairs

$$(\langle x, \emptyset\rangle =)\langle x_0, S_0\rangle, \langle x_1, S_1\rangle, \langle x_2, S_2\rangle, \ldots \tag{4}$$

form an infinite path in $\hat{\mathcal{K}}$.

For the right-to-left implication, assume any infinite sequence as in (4), for some x_i and S_i such that the condition (3) holds for every $i \in \mathbb{N}$. We construct sequences

$$y_0, y_1, \ldots \in K \quad T_0, T_1, \ldots \subseteq \mathbb{A} \quad \pi_1, \pi_2, \ldots \in \mathrm{Aut}(\mathbb{A})$$

by simultaneous induction as follows:

- $y_0 = x_0$ and $T_0 = S_0$,
- π_{i+1} is an atom automorphism such that:
  - $\pi_{i+1}(a) = a$ for $a \in \mathrm{supp}(y_i)$, and
  - $\pi_{i+1}(a) \notin \bigcup_{j=0}^{i} \mathrm{supp}(y_j)$ for $a \in \mathrm{supp}(x_{i+1} \cdot \pi_1\pi_2 \cdots \pi_i) \setminus \mathrm{supp}(y_i)$,

  and acting in an arbitrary way on the remaining atoms,
- $y_{i+1} = x_{i+1} \cdot \pi_1\pi_2 \cdots \pi_i\pi_{i+1}$,
- $T_{i+1} = S_{i+1} \cdot \pi_1\pi_2 \cdots \pi_i\pi_{i+1}$.

Notice that, since $\langle x_{i+1}, S_{i+1}\rangle$ is a legal state in $\hat{K}$, by equivariance so is $\langle y_{i+1}, T_{i+1}\rangle$. Moreover,

$$\langle\langle y_i, T_i\rangle, \langle y_{i+1}, T_{i+1}\rangle\rangle \in \hat{R}$$

for each $i \in \mathbb{N}$, therefore the sequence

$$\langle y_0, T_0\rangle, \langle y_1, T_1\rangle, \langle y_2, T_2\rangle, \dots$$

forms an infinite path in $\hat{K}$. To see this, note that (by equivariance of $\hat{R}$)

$$\langle\langle y_i \cdot \pi_{i+1}, T_i \cdot \pi_{i+1}\rangle, \langle y_{i+1}, T_{i+1}\rangle\rangle \in \hat{R}, \qquad y_i \cdot \pi_{i+1} = y_i \quad \text{and} \quad T_i \cdot \pi_{i+1} = T_i$$

since π_{i+1} by definition fixes $\mathrm{supp}(y_i)$ and $T_i \subseteq \mathrm{supp}(y_i)$.

As a consequence, the sequence $y_0, y_1, y_2, \dots$ forms a path in $\mathcal{K}$. A useful property of this path, easy to infer from the definition of y_i, is that:

$$(\mathrm{supp}(y_{i+1}) \setminus \mathrm{supp}(y_i)) \cap \bigcup_{j=0}^{i} \mathrm{supp}(y_j) = \emptyset. \tag{5}$$

In words, whenever a locally fresh atom appears in some y_i, then it does not appear anywhere earlier in the path.

We shall show that no predicate p_a holds on this path more than once. Assume towards a contradiction that $a \in \mathrm{pred}(y_i) \cap \mathrm{pred}(y_j)$ for some $a \in \mathbb{A}$ and $i < j$. Then obviously $a \in \mathrm{supp}(y_i)$ and $a \in \mathrm{supp}(y_j)$, and by induction on the difference $j - i$, using (5), we get that $a \in \mathrm{supp}(y_k)$ for all k between i and j. Again by induction, and by definition of $\hat{R}$, a belongs to all sets $T_{i+1}, T_{i+2}, \dots, T_j$. But this means that $a \in T_j \cap \mathrm{pred}(y_j)$, which contradicts the fact that $\langle y_j, T_j\rangle$ is a legal state in $\hat{K}$. This completes the proof of the right-to-left implication, and of the entire theorem.

The Power of the Filtration Technique for Modal Logics with Team Semantics

Martin Lück

Institut für Theoretische Informatik, Leibniz Universität Hannover, Hannover, Germany
lueck@thi.uni-hannover.de

Abstract

Modal Team Logic (MTL) extends Väänänen's Modal Dependence Logic (MDL) by Boolean negation. Its satisfiability problem is decidable, but the exact complexity is not yet understood very well. We investigate a model-theoretical approach and generalize the successful filtration technique to work in team semantics. We identify an "existential" fragment of MTL that enjoys the exponential model property and is therefore, like Propositional Team Logic (PTL), complete for the class AEXP(poly). Moreover, superexponential filtration lower bounds for different fragments of MTL are proven, up to the full logic having no filtration for any elementary size bound. As a corollary, superexponential gaps of succinctness between MTL fragments of equal expressive power are shown.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases dependence logic, team logic, modal logic, finite model theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.31

1 Introduction

Team semantics, introduced by Hodges [12], enjoys striking success as a compositional semantics for logics with incomplete information, like independence-friendly logic by Hintikka and Sandu [11]. In this vein, Väänänen [17] introduced *dependence logic* as an extension of first-order logic. At the heart of this extension is the atomic formula of *dependence*, written $=(x, y)$, stating that in a set (*team*) of assignments to the variables x and y, the value of y is a function of x. Related concepts are the atoms of *independence* $x \perp y$, *inclusion* $x \subseteq y$ and more [6, 7]. A rich family of logics of dependence and independence has developed, and dependence logic and its variants have found a broad range of applications like database theory, quantum mechanics and statistics.

The concept of team semantics has been introduced into other logics as well, like quantified Boolean logic [9] and modal logic [16]. A noticeable feature of this semantics is the loss of Boolean negation. Re-adding it as a special connective, often written $\sim$, yields what is called *team logic* [15, 17]. Team Logic based on propositional or modal logic is comprehensive enough to even express atoms like dependence, independence and inclusion as composite formulas using $\sim$ [10]. Therefore it is desirable to classify its exact computational complexity.

We study *Modal Team Logic* ($\mathcal{MTL}$). Its model checking problem was classified as PSPACE-complete by Müller [15]. Moreover, a finite model property was shown by Kontinen et al. [13]. The complexity of its satisfiability problem is however not yet understood well.

In this article, we pursue a model-theoretic approach to that problem via the prominent *filtration* technique. Filtration turned out to be a powerful tool to prove the *exponential model property* (if a formula φ has a model, then it has a model of size $2^{c \cdot |\varphi|}$) for a wide

range of logics. Examples are modal logics [3], dynamic and temporal logics [2], and even fragments of first-order logic [8].

In Section 2, we give the necessary foundational definitions. The filtration technique for classical modal logic is generalized to $\mathcal{MTL}$, and applied in Section 3. It is however also shown unable to fully handle team-wide modalities. In Section 4, we improve the ideas of filtration to account for such modalities, and prove the exponential model property for a non-trivial fragment of $\mathcal{MTL}$, called $\mathcal{MTL}_{\mathsf{mon}}$. Nevertheless, in the final Section 5, it is shown that for full $\mathcal{MTL}$, filtration can only yield models of non-elementary size.

2 Preliminaries

Let $[m]$ denote the set $\{ 1, \ldots, m \}$ for any $m \in \mathbb{N}$. We fix a countable set $\mathcal{PS}$ of atomic propositions $p_1, p_2, \ldots$; $\mathcal{ML}$ then denotes the classical mono-modal logic, generated by the grammar

$$\alpha ::= \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Diamond\alpha \mid \Box\alpha \mid p,$$

where $p \in \mathcal{PS}$. Furthermore, we employ the usual abbreviations and define $\alpha \to \beta := \neg\alpha \vee \beta$, $\top := \alpha \to \alpha$ and $\bot := \neg\top$.

The semantics of $\mathcal{ML}$ are the usual Kripke semantics, i.e., $\mathcal{ML}$-formulas are evaluated over *Kripke structures* $\mathcal{K} = (W, R, V)$, where (W, R) is a directed graph and $V \colon \mathcal{PS} \to \mathcal{P}(W)$ is the *valuation* function.

Modal Team Logic $\mathcal{MTL}$ extends modal logic according to the following grammar, where φ denotes an $\mathcal{MTL}$ formula and α is any $\mathcal{ML}$ formula:

$$\varphi ::= {\sim}\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \alpha.$$

$\mathcal{MTL}$ is evaluated on whole *teams* of multiple (and potentially zero) worlds in a Kripke structure. Consequently, it replaces the point-wise negation $\neg$ by the *team-wise negation* $\sim$.

While classical $\mathcal{ML}$ formulas are denoted by the letters $\alpha, \beta, \gamma, \ldots$, we use $\varphi, \psi, \vartheta, \ldots$ for $\mathcal{MTL}$ formulas.

In order to generalize modal operators to teams, we define the following terms. If $T \subseteq W$ is a team in a Kripke structure (W, R, V), then its *image* is defined as $R(T) := \bigcup_{w \in T} R(w)$, where $R(w) := \{ v \in W \mid Rwv \}$. Analogously to R-successors of worlds, a team S is a *successor team* of T if $S \subseteq R(T)$ and $T \subseteq R^{-1}(S)$; that is, if every $w \in T$ has a successor in S and every $w' \in S$ has a predecessor in T.

The semantics of $\mathcal{MTL}$ is then as follows, where $\mathcal{K} = (W, R, V)$ is a Kripke structure, $T \subseteq W$, $\varphi, \psi \in \mathcal{MTL}$ and $\alpha \in \mathcal{ML}$:

$$\begin{aligned}
(\mathcal{K}, T) &\vDash \alpha && \text{iff } \forall w \in T : (\mathcal{K}, w) \vDash \alpha \text{ (in Kripke semantics)},\\
(\mathcal{K}, T) &\vDash {\sim}\varphi && \text{iff } (\mathcal{K}, T) \nvDash \varphi\\
(\mathcal{K}, T) &\vDash \varphi \wedge \psi && \text{iff } (\mathcal{K}, T) \vDash \varphi \text{ and } (\mathcal{K}, T) \vdash \psi\\
(\mathcal{K}, T) &\vDash \varphi \vee \psi && \text{iff } \exists S, U \subseteq T \text{ such that } T = S \cup U,\ (\mathcal{K}, S) \vDash \varphi, \text{ and } (\mathcal{K}, U) \vDash \psi\\
(\mathcal{K}, T) &\vDash \Diamond\varphi && \text{iff } \exists S \subseteq W \text{ such that } S \text{ is a successor team of } T \text{ and } (\mathcal{K}, S) \vDash \varphi\\
(\mathcal{K}, T) &\vDash \Box\varphi && \text{iff } R(T) \vDash \varphi
\end{aligned}$$

The most striking difference to classical logic is perhaps that disjunction does not denote a truth-functional operation, but instead an existential quantification of subteams of the current team. We define the *Boolean* or *truth-functional disjunction* $\varphi_1 \otimes \varphi_2 := {\sim}({\sim}\varphi_1 \wedge {\sim}\varphi_2)$. The

operator $\varotimes$ is also sometimes called *intuitionistic* or *classical disjunction* in the context of team logic.[1]

The above semantics being well-defined on $\mathcal{ML}$ formulas (that is, on $\sim$-free formulas) is due to the *flatness property* of $\mathcal{ML}$ [16]. For instance, if every point $w \in T$ satisfies either p_1 or p_2 or both, and consequently $T \vDash p_1 \vee p_2$, then T has a division into T_1 satisfying p_1 and T_2 satisfying p_2, and vice versa. In general, the standard $\mathcal{ML}$ operators $(\wedge, \vee, \neg, \Diamond, \Box)$ can be evaluated locally, i.e., point-wise. When however containing the Boolean negation $\sim$ on the level of teams, then it is not possible to break down the semantics to local evaluation and the above semantics applies. This is in contrast to $\neg$, which is always applied locally and cannot contain a nested $\sim$; neither $\neg$ or $\sim$ can express the respective other one.

To express that at least one world w in a team T satisfies a given $\mathcal{ML}$ formula α (dually to the first line of the above definition), then we write $\mathsf{E}\alpha$, which can be defined as $\sim\neg\alpha$.

For $\Phi \subseteq \mathcal{MTL}$, let $\bigcirc \Phi := \{\bigcirc\varphi \mid \varphi \in \Phi\}$ for $\bigcirc \in \{\neg, \sim, \Box, \Diamond\}$.

For satisfaction we use the standard notation. $(\mathcal{K}, T) \vDash \Phi$ means that $(\mathcal{K}, T)$ satisfies every $\varphi \in \Phi$ in the above $\mathcal{MTL}$ semantics. $(\mathcal{K}, T)$ is then a *model* of Φ (or simply of φ in case $\Phi = \{\varphi\}$). Similarly, for sets $\Gamma \subseteq \mathcal{ML}$, $(\mathcal{K}, w)$ is called a model of Γ if it satisfies Γ in the standard semantics of modal logic.

$\Phi \vDash \Psi$ means that Φ entails all formulas in Ψ, i.e., any model of Φ is a model of every formula in Ψ. We also simply write $\varphi \vDash \psi$ instead of $\{\varphi\} \vDash \{\psi\}$. Likewise, if two formulas φ, ψ have the same models, then they are *equivalent*, written $\varphi \equiv \psi$.

If the structure $\mathcal{K}$ is clear, then we simply write $T \vDash \varphi$ or $w \vDash \alpha$ instead of $(\mathcal{K}, T) \vDash \varphi$ and $(\mathcal{K}, w) \vDash \alpha$.

The modality-free fragment of $\mathcal{MTL}$ is called $\mathcal{PTL}$; its satisfiability and validity problems are AEXPTIME(poly)-complete [9], where the class AEXPTIME(poly) contains all languages that are decided by an alternating Turing machine with exponential runtime bound and polynomial alternation bound (see also Chandra et al. [4]).

2.1 Morphism and Filtrations

▶ **Definition 1** (Modal Homomorphism). Let $\mathcal{K} = (W, R, V)$ and $\mathcal{K}' = (W', R', V')$ be Kripke structures. A mapping $h\colon W \to W'$ is a *homomorphism*, in symbols $h\colon \mathcal{K} \to \mathcal{K}'$, if

1. for all $p \in \mathcal{PS}$, if $w \in V(p)$, then $h(w) \in V'(p)$
2. for all $w, v \in W$, if Rwv, then $R'f(w)f(v)$.

If h is additionally surjective, then $\mathcal{K}'$ is called the *morphic image* of $\mathcal{K}$ and denoted $h(\mathcal{K})$.

If $\approx$ is an equivalence relation on a set S, then for every $s \in S$, $[s]_\approx := \{\, s' \in S \mid s' \approx s \,\}$ denotes the *equivalence class* of s in S. The set of all equivalence classes in S is the *quotient* $S/_\approx := \{\, [s]_\approx \mid s \in S \,\}$, and the *index* of $\approx$ is defined as the cardinality $|S/_\approx|$. If $U \subseteq S$, then $[U]_\approx := \{\, [s]_\approx \mid s \in U \,\}$. We often will drop the index and write $[s]$ and $[U]$. If $\approx$ and $\approx'$ are equivalence relations on S such that $s \approx' s'$ implies $s \approx s'$, then $\approx'$ is a *refinement* of $\approx$. Given two equivalence relations $\approx_1, \approx_2$ on S, their intersection $\approx_1 \cap \approx_2$ is again an equivalence relation on S and a refinement of both $\approx_1$ and $\approx_2$, and $|S/_{\approx_1 \cap \approx_2}| \leq |S/_{\approx_1}| \cdot |S/_{\approx_2}|$.

[1] The original term used by Väänänen was in fact "Boolean disjunction" [16, 17]. The synonymous adjective "classical" apparently stems from the disjunction acting truth-functionally and hence just as known from classical (point-wisely evaluated) logic. It is however potentially confusing, since already a simple expression like $p \varotimes q$ is neither a classical formula, nor semantically equivalent to one. The term "intuitionistic disjunction" stems from a work of Abramsky and Väänänen [1] on connections between linear logic and team logic, but is more commonly found in linear logic. Of all mentioned terms, "Boolean disjunction" is probably the least confusing.

▶ **Definition 2** (Filtration). Let $\mathcal{K} = (W, R, V)$ be a Kripke structure. Let $\approx$ be an equivalence relation on W. Then the Kripke structure (W', R', V') defined by

$$W' := W/_{\approx}, \tag{1}$$
$$R'[w][v] \Leftrightarrow \exists w' \in [w], \exists v' \in [v] \text{ such that } Rwv, \tag{2}$$
$$[w] \in V'(p) \Leftrightarrow \exists w' \in [w] \cap V(p), \tag{3}$$

is a *filtration of* $\mathcal{K}$ *through* $\approx$, denoted $\mathcal{K}/_{\approx}$.[2]

Clearly every filtration of a structure is also a morphic image of it, via the mapping $w \mapsto [w]$.

Standard Modal Logic allows, for any given formula $\alpha \in \mathcal{ML}$ and model $\mathcal{K}$, filtration down to a model of α of size exponential in α. The approach is the following: For fixed Kripke structures and a subset $\Gamma \subseteq \mathcal{ML}$, we define an equivalence relation $\approx_\Gamma$ of "agreement", namely $w \approx_\Gamma w'$ if and only if $\forall \alpha \in \Gamma : (\mathcal{K}, w) \vDash \alpha \Leftrightarrow (\mathcal{K}, w') \vDash \alpha$.

The *subformulas* of (sets of) $\mathcal{MTL}$ formulas are defined inductively:

$$\begin{aligned}
\mathrm{SF}(p) &:= \{p\} && \text{if } p \in \mathcal{PS},\\
\mathrm{SF}(\triangle\varphi) &:= \{\triangle\varphi\} \cup \mathrm{SF}(\varphi) && \text{if } \triangle \in \{\neg, \sim, \Box, \Diamond\},\\
\mathrm{SF}(\varphi \circ \psi) &:= \{\varphi \circ \psi\} \cup \mathrm{SF}(\varphi) \cup \mathrm{SF}(\psi) && \text{if } \circ \in \{\wedge, \vee\},\\
\mathrm{SF}(\Phi) &:= \bigcup_{\varphi \in \Phi} \mathrm{SF}(\varphi) && \text{if } \Phi \subseteq \mathcal{MTL}.
\end{aligned}$$

▶ **Theorem 3** ([3]). *Let $\mathcal{K} = (W, R, V)$ be a Kripke structure, and let $\Gamma \subseteq \mathcal{ML}$ be closed under taking subformulas, i.e., $\mathrm{SF}(\Gamma) = \Gamma$. Let $\approx'$ be a refinement of $\approx_\Gamma$.*

Then $(\mathcal{K}, w) \vDash \alpha$ iff $(\mathcal{K}/_{\approx}, [w]_{\approx'}) \vDash \alpha$, for all formulas $\alpha \in \Gamma$ and worlds $w \in W$.

Proof. Proven identically to [3, Theorem 2.39].[3] ◀

▶ **Corollary 4** (Small Model Property of Modal Logic). *Every satisfiable formula $\alpha \in \mathcal{ML}$ has a model of size at most $2^{|\alpha|}$.*

3 Filtration in Team Semantics

An obvious generalization of Theorem 3 is to replace the quantification "every world w in $\mathcal{K}$" by "every team T in $\mathcal{K}$". It is plausible that a similar inductive proof works for teams. The next definition is a first attempt: It permits to prove certain filtration results for team semantics, but its limits are quickly reached, as this section later demonstrates.

▶ **Definition 5.** If $\approx$ is an equivalence relation on a Kripke structure $\mathcal{K} = (W, R, V)$, $T \subseteq W$ is a team, and $\Phi \subseteq \mathcal{MTL}$, then $\approx$ is

Φ-*invariant on* $(\mathcal{K}, T)$ if $\forall \varphi \in \Phi : (\mathcal{K}, T) \vDash \varphi \Leftrightarrow (\mathcal{K}/_{\approx}, [T]_{\approx}) \vDash \varphi$,

Φ-*invariant on* $\mathcal{K}$ if it is $(\mathcal{K}, T)$-invariant for all $T \subseteq W$.

strongly Φ-*invariant* on $\mathcal{K}$ (resp. $(\mathcal{K}, T)$) if every refinement $\approx'$ of $\approx$ is Φ-invariant on $\mathcal{K}$ (resp. $(\mathcal{K}, T)$).

---

[2] The definition of R' used here is also known of the *minimal filtration* (of R), but all results in this paper can be proven for filtrations with a larger number of edges.

[3] The proof considers only $\approx' = \approx_\Gamma$. Nevertheless, it completely goes through if the first characterizing property of a filtration, namely $W' := \{\, [w] \mid w \in W \,\}$, is relaxed, as long as the mapping $[\cdot]\colon w \mapsto [w]$ respects (2) and (3) in Definition 2.

▶ **Proposition 6.** *Let $\Gamma \subseteq \mathcal{ML}$. Then on any structure $\mathcal{K}$, the corresponding equivalence relation $\approx_{\mathrm{SF}(\Gamma)}$ is strongly Γ-invariant on $\mathcal{K}$.*

Proof. Let $\mathcal{K} = (W, V, R)$, and let $\approx'$ be a refinement of $\approx_{\mathrm{SF}(\Gamma)}$. We have to show that $\approx'$ is α-invariant on $\mathcal{K}$ for all $\alpha \in \Gamma$. By Theorem 3, it holds $w \vDash \alpha \Leftrightarrow [w]_{\approx'} \vDash \alpha$ for all $w \in W$. The statement is then proven, since for all $T \subseteq W$,

$$
\begin{aligned}
& T \vDash \alpha \\
\Leftrightarrow\ & \forall\, w \in T : w \vDash \alpha && \text{(since } \alpha \in \mathcal{ML}\text{)} \\
\Leftrightarrow\ & \forall\, w \in T : [w]_{\approx'} \vDash \alpha && \text{(by assumption)} \\
\Leftrightarrow\ & \forall\, [w]_{\approx'} \in [T]_{\approx'} : [w]_{\approx'} \vDash \alpha && \text{(by definition of } [T]_{\approx'}\text{)} \\
\Leftrightarrow\ & [T]_{\approx'} \vDash \alpha && \text{(since } \alpha \in \mathcal{ML}\text{).}
\end{aligned}
$$

◀

The above result demonstrates that team semantics has filtration when restricted to "flat" formulas. However, not all $\mathcal{MTL}$ formulas feature the flatness property. We proceed with fragments of $\mathcal{MTL}$ which are strictly stronger than $\mathcal{ML}$, and show that they still inherit the property to admit filtration.

▶ **Definition 7** ($\mathcal{B}$- and $\mathcal{S}$-closures)**.** If $\Phi \subseteq \mathcal{MTL}$, then $\mathcal{B}(\Phi)$ denotes the closure of Φ under $\sim$ and $\wedge$. Moreover, $\mathcal{S}(\Phi)$ denotes the closure of Φ under $\sim, \wedge$ and $\vee$.

Clearly $\Phi \subseteq \mathcal{B}(\Phi) \subseteq \mathcal{S}(\Phi) \subseteq \mathcal{MTL}$. However, every $\mathcal{MTL}$ formula is already expressively equivalent to even a $\mathcal{B}(\mathcal{ML})$ formula [13], a property that the recent axiomatization of $\mathcal{MTL}$ [14] allows to prove for each formula syntactically.

▶ **Lemma 8.** *Let $\mathcal{K} = (W, R, V)$ be a structure and $\Phi \subseteq \mathcal{MTL}$. If $\approx$ is strongly Φ-invariant on $\mathcal{K}$, then $\approx$ is also strongly $\mathcal{S}(\Phi)$-invariant on $\mathcal{K}$.*

Proof. Let $\approx'$ be a refinement of $\approx$. The proof is by induction on $|\varphi|$, where $\varphi \in \mathcal{S}(\Phi)$. The inductive step for the truth-functional connectives on the level of teams, i.e., $\sim$ and $\wedge$, are clear. So assume $\varphi = \psi_1 \vee \psi_2$.

Suppose $(\mathcal{K}, T) \vDash \varphi$ via $S \cup U = T$, $S \vDash \psi_1$ and $U \vDash \psi_2$. Then $[T]_{\approx'} = [S]_{\approx'} \cup [U]_{\approx'}$. By induction hypothesis, $[T]_{\approx'} \vDash \psi_1 \vee \psi_2$.

Conversely, let $[T]_{\approx'} \vDash \varphi$ via teams $\tilde{S}, \tilde{U} \subseteq W/_{\approx'}$ such that $\tilde{S} \cup \tilde{U} = [T]_{\approx'}$, $\tilde{S} \vDash \psi_1$ and $\tilde{U} \vDash \psi_2$. There is not necessarily a unique choice of $S, U \subseteq T$ such that $[S]_{\approx'} = \tilde{S}$ and $[U]_{\approx'} = \tilde{U}$, so we choose corresponding S and U as large as possible to ensure T is covered by $S \cup U$. Namely, define $S := \{\, w \in T \mid [w]_{\approx'} \in \tilde{S} \,\}$ and $U := \{\, w \in T \mid [w]_{\approx'} \in \tilde{U} \,\}$. If now $w \in T$, then $[w]_{\approx'} \in [T]_{\approx'}$, so $[w]_{\approx'} \in \tilde{S}$ or $[w]_{\approx'} \in \tilde{U}$, and consequently $w \in S$ or $w \in U$. Therefore $T \subseteq S \cup U$. By definition $S, U \subseteq T$, so $T = S \cup U$.

To show $T \vDash \psi_1 \vee \psi_2$ applying the induction hypothesis, it remains to show that actually $[S]_{\approx'} = \tilde{S}$ resp. $[U]_{\approx'} = \tilde{U}$ holds. Suppose $[w]_{\approx'} \in [S]_{\approx'}$. (The proof is analogous for U). Then by definition of $[\cdot]_{\approx'}$ there exists $\hat{w} \in S$ such that $\hat{w} \approx' w$, again implying by definiton of S that $[\hat{w}]_{\approx'} = [w]_{\approx'} \in \tilde{S}$. Hence $[S]_{\approx'} \subseteq \tilde{S}$.

Let conversely $[w]_{\approx'} \in \tilde{S}$. Then $[w]_{\approx'} \cap T$ is non-empty, since otherwise $[w]_{\approx'} \notin [T]_{\approx'}$ by definition of $[\cdot]_{\approx'}$, contradicting $[w]_{\approx'} \in \tilde{S} \subseteq [T]_{\approx'}$. Hence there exists some $\hat{w} \in T$ such that $\hat{w} \approx' w$. Since $\hat{w} \in T$ and $[\hat{w}]_{\approx'} = [w]_{\approx'} \in \tilde{S}$, it follows $\hat{w} \in S$ by the definition of S, and hence $[\hat{w}]_{\approx'} = [w]_{\approx'} \in [S]_{\approx'}$ as desired. ◀

▶ **Theorem 9.** *For every Kripke structure $\mathcal{K}$ and every finite $\Phi \subseteq \mathcal{S}(\mathcal{ML})$ there is an equivalence relation of index at most $\prod_{\varphi \in \Phi} 2^{|\varphi|}$ that is strongly Φ-invariant on $\mathcal{K}$.*

Proof. Let $\Gamma := \mathrm{SF}(\Phi) \cap \mathcal{ML}$. By Proposition 6, $\approx_\Gamma$ is Γ-invariant. $|\Gamma| \leq \sum_{\varphi\in\Phi} |\varphi|$, so $\approx_\Gamma$ has index at most $\prod_{\varphi\in\Phi} 2^{|\varphi|}$. Since $\Phi \subseteq \mathcal{S}(\Gamma)$, the theorem follows from Lemma 8. ◀

▶ **Corollary 10.** *Every satisfiable $\varphi \in \mathcal{S}(\mathcal{ML})$ has a model of size at most $2^{|\varphi|}$.*

3.1 Lower bounds for filtrations with invariance

In the context of modal logic, (strong) invariance appears as a natural property of filtrations. It seems like a straightforward tool to generalize the usual filtration technique, which preserves the truth of certain formulas in all *points* of a model, to team semantics. But it is inappropriate when team-wide modalities come into play, as the following counter-example shows. It employs a $\mathcal{PTL}$ formula of Hannula et al. [10], namely

$$\max(\Phi) := \sim \bigvee_{p\in\Phi} (p \mathbin{\mathbb{O}} \neg p)\,,$$

where $\Phi \subseteq \mathcal{PS}$ is a finite set of propositions. Clearly $\max(\Phi)$ has length $\mathcal{O}(|\Phi|)$. Intuitively, it is true in a team T if and only if all Boolean assignments to variables in Φ are "realized" in worlds of T, formally:

$$\begin{aligned}(W,R,V,T) \vDash \max(\Phi) \Leftrightarrow \quad & \{\, f_w : \Phi \to \{1,0\} \mid w \in W, f_w(p) = 1 \Leftrightarrow w \in V(p) \,\} \\ & = \{\, f \mid f : \Phi \to \{1,0\} \,\}\end{aligned}$$

It is equivalent to the formula

$$\bigwedge_{\Phi'\in\mathcal{P}(X)} \mathsf{E}\left(\bigwedge_{p\in\Phi'} p \wedge \bigwedge_{p\in\Phi\setminus\Phi'} \neg p\right).$$

In the counter-example, we more generally consider homomorphisms that feature similar invariance properties as equivalence relations.

▶ **Definition 11.** Let $h\colon \mathcal{K} \to \mathcal{K}'$ be a homomorphism between Kripke structures. If T is a team in $\mathcal{K}$, then $h(T) := \{\, h(w) \mid w \in T \,\}$.

h is called Φ-*invariant on* $(\mathcal{K}, T)$ if $(\mathcal{K}, T) \vDash \varphi \Leftrightarrow (\mathcal{K}', h(T)) \vDash \varphi$ for all $\varphi \in \Phi$.

h is called Φ-*invariant on* $\mathcal{K}$ if it is Φ-invariant on $(\mathcal{K}, T)$ for all teams T in $\mathcal{K}$.

In the following, we also simply write that a homomorphism or equivalence relation is φ-invariant (resp. φ-preserving) instead of $\{\varphi\}$-invariant (resp. $\{\varphi\}$-preserving).

▶ **Theorem 12.** *Suppose $\varphi = \Box\max(\Phi)$ for $\Phi \subseteq \mathcal{PS}$. Then there is a structure $\mathcal{K}$ such that the image of any homomorphism φ-invariant on $\mathcal{K}$ has size at least $2^{2^{|\Phi|}}$.*

Proof. Let $\Phi := \{p_1, \ldots, p_n\} \subseteq \mathcal{PS}$, and let $\varphi := \Box\max(\Phi)$. Construct the structure $\mathcal{K} = (W, R, V)$ as follows. Let $W := \mathcal{P}(\mathcal{P}(\Phi))$, consequently $|W| = 2^{2^n}$. Intuitively, every world $\mathcal{A} \in W$ corresponds to a (possibly empty) subset of Boolean assignments to $p_1, \ldots, p_n$, each assignment being represented by a subset of Φ.

For any set $\mathcal{A} = \{\Phi'\} \in W$ of exactly one assignment $\Phi' \subseteq \Phi$, let $\mathcal{A} \in V(p) \Leftrightarrow p \in \Phi'$, that is, worlds in the Kripke structure that are singletons mimic the propositional labeling represented by their unique member Φ'. In all other worlds, all propositions are true, i.e., if $|\mathcal{A}| \neq 1$, then $\mathcal{A} \in V(p)$ for all $p \in \Phi$. Finally, for all $\mathcal{A} \in W$ and $\Phi' \in \mathcal{A}$, add the edge from $\mathcal{A}$ to $\{\Phi'\}$.

Let now $h(\mathcal{K})$ be a morphic image of $\mathcal{K}$ with h being φ-invariant. By definition, h has to preserve the truth of all propositions $p \in \Phi$. However, it also has to preserve their *falsity* everywhere: Otherwise there is some $\subseteq$-minimal $\Phi' \subseteq \Phi$ such that $\{\Phi'\} \not\models p$, but $h(\{\Phi'\}) \models p$. Then the assignment Φ' itself cannot occur in $h(\mathcal{K})$ anymore, in particular, $h(\mathcal{K})$ contains no world w such that $w \not\models p$ iff $p \notin \Phi'$. Then $\sim\varphi$ holds in all teams of $h(\mathcal{K})$ despite $(\mathcal{K}, W) \models \varphi$, contradiction to the φ-invariance.

Consequently, h preserves truth and falsity of propositions $p \in \Phi$. Suppose that the image $h(\mathcal{K})$ has *less* than 2^{2^n} worlds. By cardinality constraints, h is not injective, i.e., $h(\mathcal{A}) = h(\mathcal{A}')$ for distinct $\mathcal{A}, \mathcal{A}' \in W$. W.l.o.g. there is an assignment $\Phi' \in \mathcal{A} \setminus \mathcal{A}'$. Consider now the team $T := \{\, \mathcal{A}', \mathcal{P}(\Phi) \setminus \Phi' \,\}$ where neither world has $\{\Phi'\}$ as successor. Hence $T \not\models \Box \max(\Phi)$.

But by definition, the second world in T has $\{\Phi''\}$ as successor for any assignment $\Phi'' \subseteq \Phi$ except Φ'. Note that

$$h(T) = \{h(\mathcal{A}'), h(\mathcal{P}(\Phi) \setminus \{\Phi'\})\} = \{h(\mathcal{A}), h(\mathcal{P}(\Phi) \setminus \{\Phi'\})\}.$$

As h preserves edges, for every $\Phi'' \subseteq \Phi$ an element in $h(T)$ has $h(\{\Phi''\})$ as a successor: for $\Phi'' = \Phi'$ it is $h(\mathcal{A})$, and for $\Phi'' \neq \Phi'$ it is $h(\mathcal{P}(\Phi) \setminus \Phi')$. Furthermore, for distinct singletons $\{\Phi''\}, \{\Phi'''\} \in W$ always $h(\{\Phi''\}) \neq h(\{\Phi'''\})$, as otherwise at least one proposition would not be preserved. Hence $h(T) \models \Box \max(\Phi)$. But as $T \not\models \Box \max(\Phi)$, h cannot be φ-invariant. ◀

▶ **Corollary 13.** *There are $\mathcal{MTL}$ formulas φ and structures $\mathcal{K}$ such that every φ-invariant equivalence relation on $\mathcal{K}$ has index $2^{2^{\Omega(|\varphi|)}}$.*

Moreover, the theorem exhibits a gap between $\mathcal{B}(\mathcal{ML})$ and $\mathcal{S}(\mathcal{ML})$ with respect to distributing modal operators. Recall that according to Theorem 9, $\mathcal{S}(\mathcal{ML})$ admits exponential filtration.

▶ **Corollary 14.** *There are formulas $\Box\psi$, where $\psi \in \mathcal{S}(\mathcal{ML})$, such that every $\mathcal{S}(\mathcal{ML})$ formula equivalent to $\Box\psi$ has length $2^{\Omega(|\psi|)}$.*

In contrast, if $\varphi \in \mathcal{B}(\mathcal{ML})$, then $\Box\varphi$ has an equivalent $\mathcal{B}(\mathcal{ML})$ formula already of length $\leq 2|\varphi|$. The translation is immediate, since $\Box{\sim}\psi \equiv {\sim}\Box\psi$ and $\Box(\psi \wedge \psi') \equiv \Box\psi \wedge \Box\psi'$. In other words, $\Box$ *is easy to distribute over* $\wedge$ *and* $\sim$*, but hard to distribute over* $\vee$.

▶ **Corollary 15.** *There are formulas $\psi \in \mathcal{S}(\mathcal{ML})$ such that every $\mathcal{B}(\mathcal{ML})$ formula equivalent to ψ has length $2^{\Omega(|\psi|)}$.*

On the other hand, $\Diamond$ is easy to distribute over $\vee$, as $\Diamond(\psi \vee \psi') \equiv \Diamond\psi \vee \Diamond\psi'$.

4 Weaker filtrations for monotone MTL

An exponential model property can be obtained for larger fragments of $\mathcal{MTL}$, provided the requirements of filtration are weakened properly. An obvious candidate is the *invariance* property. To find a small model of φ starting from a given model $(\mathcal{K}, T)$, it is unnecessary to have $\sim\varphi$ preserved as well; hence we replace invariance by asymmetric preservation.

Moreover, a filtration $\approx$ does not need to preserve a formula φ in *all* teams of a model $(\mathcal{K}, T)$ — having φ true in $[T]_\approx$ would be completely sufficient. For this reason, we do not define preservation on the whole structure $\mathcal{K}$, but only locally:

▶ **Definition 16.** If $\approx$ is an equivalence relation on a Kripke structure $\mathcal{K} = (W, R, V)$, $T \subseteq W$ is a team, and $\Phi \subseteq \mathcal{MTL}$, then $\approx$ is

Φ-*preserving on* $(\mathcal{K}, T)$ if $\forall\varphi \in \Phi : (\mathcal{K}, T) \models \varphi \;\Rightarrow\; (\mathcal{K}/_\approx, [T]_\approx) \models \varphi$,

strongly Φ-*preserving* if every refinement $\approx'$ of $\approx$ is Φ-preserving.

Of course Φ-preservation does not imply $\sim\Phi$-preservation. The property is however still closed under application of *monotone* connectives. In this context, we consider the $\mathcal{MTL}$ operators $\wedge, \vee, \Diamond$ and $\Box$ as monotone, and also add the Boolean disjunction $\mathbin{⫽}$. Accordingly, we define the following fragment.

▶ **Definition 17.** The fragment $\mathcal{MTL}_{\mathsf{mon}}$ of $\mathcal{MTL}$ is defined as the closure of $\mathcal{S}(\mathcal{ML})$ under $\wedge, \mathbin{⫽}, \vee, \Box$ and $\Diamond$.

▶ **Theorem 18.** *For every finite $\Phi \subseteq \mathcal{MTL}_{mon}$, every structure $\mathcal{K} = (W, R, V)$, and every team $T \subseteq W$, there is an equivalence relation of index at most $\prod_{\varphi\in\Phi} 2^{|\varphi|}$ that is strongly Φ-preserving on $(\mathcal{K}, T)$.*

Note that we still quantify over all teams T, but are allowed to choose a different filtration for each team. The order of these quantifications makes a crucial difference here, in particular it eliminates the vulnerability against the method of Theorem 12 for filtration lower bounds.

In the following auxiliary lemmas, let $\mathcal{K} = (W, V, R)$ be a Kripke structure and $\approx$ an equivalence relation on W, and accordingly $\mathcal{K}/_{\approx} = (W/_{\approx}, R', V')$ as in Definition 2.

First we prove that subformulas starting with $\Diamond$ are preserved.

▶ **Lemma 19.** *If S is a successor team of T, then $[S]_{\approx}$ is an successor team of $[T]_{\approx}$.*

Proof. Suppose that S is a successor team of T. We have to show that every $[w] \in [T]$ has an R'-successor in $[S]$ and that every $[v] \in [S]$ has an R'-predecessor in $[T]$. Let $[w] \in [T]$, then $w' \in T$ for some $w' \approx w$. w' has an R-successor $v \in S$, so $[v] \in [S]$. But $Rw'v$ implies $R'[w'][v]$, so $[w'] = [w]$ has an R'-successor in $[S]$.

Conversely, if $[v] \in [S]$, then $v' \in S$ for some $v' \approx v$. v' has an R-predecessor $w \in T$. However, Rwv' again implies $R'[w][v']$, so $R'[w][v]$ for some $[w] \in [T]$. ◀

Formulas starting with $\Box$ are similarly preserved in teams T, at least as long as the filtration does not cross the boundaries of the preimage team T:

▶ **Lemma 20.** *If S is the image of T, and $w \approx w'$ implies $w \in T \Leftrightarrow w' \in T$, then $[S]$ is the image of $[T]$.*

Proof. As in the previous lemma, every $[v] \in [S]$ has an R'-predecessor in $[T]$. It remains to prove that $[S]$ contains all R'-successors $[v]$ of all $[w] \in [T]$. Let $R'[w][v]$ for $[w] \in [T]$. There exist $w' \approx w$ and $v' \approx v$ such that $Rw'v'$. By assumption of the lemma, $w' \in T$, so its R-successor v' must be in S, and $[v'] = [v] \in [S]$. ◀

It is easy to verify that the converse of the above lemmas is not true. Moreover, we are now ready to prove the theorem.

Proof of Theorem 18. Let $\mathcal{K} = (W, R, V)$, $T \subseteq W$ and $\Phi \subseteq \mathcal{MTL}_{\mathsf{mon}}$ be as in Theorem 18. W.l.o.g. $(\mathcal{K}, T) \vDash \Phi$. For this reason, we simply show that $\varphi := \bigwedge_{\psi\in\Phi} \psi$ is strongly preserved.

By definition of $\mathcal{MTL}_{\mathsf{mon}}$, φ is a monotone combination (i.e., using only operators $\wedge, \mathbin{⫽}, \vee, \Diamond, \Box$) of $\mathcal{S}(\mathcal{ML})$ formulas. We exploit the monotonicity and define a *witness set* $\mathcal{T}(\psi)$ for certain subformulas ψ of φ. In the proof it suffices to preserve these subformulas in their corresponding witness teams instead of the whole structure.

For a simpler proof, we assume w.l.o.g. that every subformula of φ occurs only once in φ.

$\mathcal{T}(\psi) \subseteq W$ is defined in top-down manner, for $\psi \notin \mathcal{S}(\mathcal{ML})$, such that $(\mathcal{K}, \mathcal{T}(\psi)) \vDash \psi$. Accordingly, $\mathcal{T}(\varphi) := T$. Whenever $\mathcal{T}(\Box\psi)$ is defined for $\Box\psi \in \mathrm{SF}(\varphi)$, set $\mathcal{T}(\psi) := R(\mathcal{T}(\Box\psi))$. Similarly, $\mathcal{T}(\Diamond\psi)$ must have a successor team S that satisfies ψ, so set $\mathcal{T}(\psi) := S$. Any team $\mathcal{T}(\psi \vee \psi')$, for $\psi \vee \psi' \in \mathrm{SF}(\varphi)$, likewise can be split into $S \vDash \psi$ and $U \vDash \psi'$, consequently

then $\mathcal{T}(\psi) := S$ and $\mathcal{T}(\psi') := U$. If $\psi = \psi' \wedge \psi''$ or $\psi = \psi' \mathbin{⫽} \psi''$ is in $\mathrm{SF}(\varphi)$, then $\mathcal{T}(\psi')$ and/or $\mathcal{T}(\psi'')$ simply equals $\mathcal{T}(\psi)$.

Similarly as in Theorem 9, $\Gamma := \mathrm{SF}(\varphi) \cap \mathcal{ML}$. Define $\approx'$ as the coarsest refinement of $\approx_\Gamma$ that does not cross the boundaries of witness teams $\mathcal{T}(\psi)$, i.e., $w \approx' w'$ if and only if $w \approx_\Gamma w'$ and, for all $\psi \in \mathrm{SF}(\varphi)$, $w \in \mathcal{T}(\psi) \Leftrightarrow w' \in \mathcal{T}(\psi)$.

Lemma 19 and 20 now allow to prove $(\mathcal{K}/_{\approx''}, [\mathcal{T}(\psi)]_{\approx''}) \vDash \psi$ for any refinement $\approx''$ of $\approx'$, and any $\psi \in \mathrm{SF}(\varphi)$, by induction on $|\psi|$. The splitting case is proven as in Lemma 8. As $\approx'$ has index at most $2^{|\varphi|}$, this proves the theorem. ◀

▶ **Corollary 21.** *Every satisfiable formula $\varphi \in \mathcal{MTL}_{\mathsf{mon}}$ has a model of size at most $2^{|\varphi|}$.*

▶ **Proposition 22.** *The satisfiability problem of $\mathcal{MTL}_{\mathsf{mon}}$ is* AEXPTIME(poly)*-complete.*

Proof. The hardness already holds for $\mathcal{PTL}$ [9], so we prove only the upper bound. The model checking problem for $\mathcal{MTL}$ is decidable by an alternating Turing machine that, given $(\mathcal{K}, T, \varphi)$, runs in time polynomial in $|\mathcal{K}| + |\varphi|$, and with alternations polynomial in $|\varphi|$ [10]. This allows to decide the satisfiability problem of $\mathcal{MTL}_{\mathsf{mon}}$ as follows. Given a formula φ, guess a Kripke structure $\mathcal{K}$ of size up to $2^{|\varphi|}$ and a team T in $\mathcal{K}$. Then execute the above model checking algorithm on $(\mathcal{K}, T, \varphi)$. By the preceding corollary, the algorithm decides $\mathcal{MTL}_{\mathsf{mon}}$ in exponential runtime and polynomially many alternations. ◀

5 Lower bounds for MTL with alternating modalities

The established small model property for $\mathcal{MTL}_{\mathsf{mon}}$ crucially depends on the existential quantification of successor teams and splittings that is inherent to this fragment. If Boolean negations $\sim$ are permitted in front of arbitrary $\Diamond$ and $\vee$ operators, then *alternations* between existential and universal quantification of modalities are introduced.

Indeed, we prove in this section that already $\mathcal{MTL}$ formulas of the form $\sim\Diamond\sim\Diamond\cdots\varphi$ are highly resistant to filtration for very simple φ, e.g., of $\mathcal{B}(\mathcal{ML})$. Note that every $\mathcal{MTL}$ formula is equivalent to a $\mathcal{B}(\mathcal{ML})$ formula of the form

$$\bigvee_{i=1}^{n} \left(\alpha_i \wedge \bigwedge_{j=1}^{m_i} \mathsf{E}\beta_{i,j} \right),$$

where $\alpha_i, \beta_{i,j} \in \mathcal{ML}$ [13, 14]. We refer to this form as *disjunctive normal form (DNF)*, as $⫽$ and $\wedge$ are the Boolean disjunction and conjunction over teams. $\mathcal{ML}$ formulas are then considered "literals." There is only one positive literal per disjunct as $\mathcal{ML}$ is closed under $\wedge$.

In the following, the positive literals of DNFs are w.l.o.g. pairwise distinct. This allows a concise notation: For finite $\Gamma, \Delta \subseteq \mathcal{ML}$ and $\lambda : \Gamma \to \mathcal{P}(\Delta)$, we write $(\Gamma, \Delta, \lambda)$ instead of

$$\bigvee_{\alpha \in \Gamma} \left(\alpha \wedge \bigwedge_{\beta \in \lambda(\alpha)} \mathsf{E}\beta \right).$$

Here, Γ is the set of positive literals, Δ is the set of negative literals, and λ defines the clauses.

To establish filtration lower bounds, we require the disjuncts of a DNF to be "independent enough". We formalize this with two properties.

▶ **Definition 23.** A set $\Gamma \subseteq \mathcal{ML}$ is called *multiplicative* if it is satisfiable, and furthermore for all $\Gamma_1, \Gamma_2, \Gamma_3 \subseteq \Gamma$ it holds that, if $\Gamma_1 \cup \neg\Gamma_2$ and $\Gamma_1 \cup \neg\Gamma_3$ are satisfiable, then also $\Gamma_1 \cup \neg(\Gamma_2 \cup \Gamma_3)$ is satisfiable.

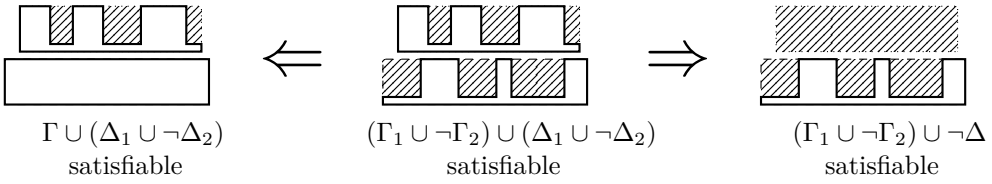


Figure 1 Hanoi property of sets $\Gamma, \Delta \subseteq \mathcal{ML}$.

Examples for multiplicative sets are $\mathcal{PS}$ and $\{ \Box\varphi \mid \varphi \in \mathcal{ML} \}$.

▶ **Definition 24.** Let $\Gamma, \Delta \subseteq \mathcal{ML}$. The tuple (Γ, Δ) has the *Hanoi property* if for all $\Gamma_1, \Gamma_2 \subseteq \Gamma$ and $\Delta_1, \Delta_2 \subseteq \Delta$:

1. $\Gamma \cup \neg\Delta$ is satisfiable,
2. whenever $\Gamma_1 \cup \neg\Gamma_2$ is satisfiable, then also $(\Gamma_1 \cup \neg\Gamma_2) \cup \neg\Delta$ is satisfiable,
3. whenever $\Delta_1 \cup \neg\Delta_2$ is satisfiable, then also $\Gamma \cup (\Delta_1 \cup \neg\Delta_2)$ is satisfiable.

For convenience, we say that a DNF $(\Gamma, \Delta, \lambda)$ has this property simply if (Γ, Δ) has it.

An example illustrating the Hanoi property[4] of a tuple (Γ, Δ) is depicted in Figure 1.

As a next step, we more carefully analyze the entailment relation $\models$ between the literals in each Γ and Δ. For the rest of the section, we make use of a number of definitions from order theory, like partial orders, lattices and filters. To refresh the foundations, the reader is referred to the very good textbook by Davey and Priestley [5].

An *antichain* is a set of pairwise unordered elements. The *width* $w(X)$ of a partially ordered finite set X is the cardinality of its largest antichain. If $S \subseteq S'$ is ordered by $\preceq$, then its *up-set* is $S^{\uparrow} := \{ s' \in S' \mid \exists s \in S : s \preceq s' \}$. Its *down-set* $S^{\downarrow}$ is defined analogously. $\mathcal{U}(X)$ resp. $\mathcal{L}(X)$ denotes the lattice of all upper resp. lower subsets of X. By convention, we assume the quasi-ordering $\models$ on $\mathcal{MTL}$, and the partial ordering $\subseteq$ on $\mathcal{P}(\mathcal{MTL})$.

Define the formula $\mathsf{NESub}\psi := \top \vee (\mathsf{E}\top \wedge \psi)$. It states that a non-empty subteam satisfies ψ. Furthermore, let $\mathsf{AS}\psi := {\sim}\mathsf{NESub}{\sim}\mathsf{NESub}\psi$.

▶ **Lemma 25.** *For any Kripke structure $\mathcal{K}$ and team T in $\mathcal{K}$, $(\mathcal{K}, T) \models \mathsf{AS}\psi$ if and only if $(\mathcal{K}, \{w\}) \models \psi$ for every singleton $\{w\} \subseteq T$.*

Proof. For the first direction, let $(\mathcal{K}, T) \models \mathsf{AS}\psi$ and $w \in T$. For the sake of contradiction suppose $(\mathcal{K}, \{w\})) \not\models \psi$. Clearly the only non-empty subteam of $\{w\}$ is $\{w\}$ itself, so no non-empty subteam of $\{w\}$ satisfying ψ exists. For this reason, $(\mathcal{K}, \{w\}) \models {\sim}\mathsf{NESub}\psi$. Clearly $\{w\}$ is a non-empty subteam of T, consequently $(\mathcal{K}, T) \models \mathsf{NESub}{\sim}\mathsf{NESub}\psi$, contradicting $(\mathcal{K}, T) \models \mathsf{AS}\psi$.

For the other direction, suppose $(\mathcal{K}, \{w\}) \models \psi$ for every singleton $\{w\} \subseteq T$. Assume again for the sake of contradiction that $(\mathcal{K}, T) \not\models \mathsf{AS}\psi$, i.e., $\mathsf{NESub}{\sim}\mathsf{NESub}\psi$ holds in T via some non-empty subteam $T' \subseteq T$ that satisfies ${\sim}\mathsf{NESub}\psi$. Since T' clearly contains at least one singleton that is also contained in T, it satisfies $\mathsf{NESub}\psi$ as well, contradiction. ◀

Using the above properties, we now identify a class of formulas in DNF in which the optimal filtration directly correlates with the size of an $\models$-antichain in the set of its positive literals.

[4] As in the *Towers of Hanoi* puzzle, the upper layer (true elements of Δ in Figure 1) should be empty unless the lower layer (true elements of Γ) is "full". The property then states that, starting at a model for any subsets of Γ and Δ, it is also possible to find models with the lower layer (Γ) changed to "full" or the upper one (Δ) changed to "empty".

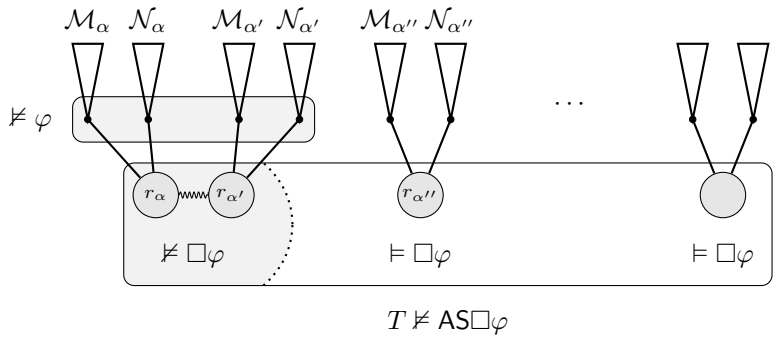


Figure 2 A filtration failing to preserve $\Box\varphi$ in the singleton $[r_\alpha] = [r_{\alpha'}]$.

▶ **Theorem 26.** *Let a DNF $\varphi = (\Gamma, \Delta, \lambda)$ satisfy the following properties:*
1. *φ has the Hanoi property and $\Gamma \cup \Delta$ is multiplicative,*
2. *Γ is partially ordered by $\models$,*
3. *$\lambda : \Gamma \to \Delta$, and $\alpha \models \alpha' \;\Leftrightarrow\; (\alpha' \wedge \lambda(\alpha')) \models (\alpha \wedge \lambda(\alpha))$.*
Then there exists a model $(\mathcal{K}, T)$ such that every equivalence relation $\approx$ that strongly preserves $\mathsf{AS}\Box\varphi$ on $(\mathcal{K}, T)$ has index at least $w(\Gamma)$.

Proof. Let $\Gamma^* \subseteq \Gamma$ be an antichain of cardinality $w(\Gamma)$. We construct $\mathcal{K}$ as follows. By multiplicativity and the Hanoi property, every $\alpha \in \Gamma$ is satisfiable by a model $\mathcal{M}_\alpha = (\mathcal{K}^{\mathcal{M}_\alpha}, w^{\mathcal{M}_\alpha})$ such that

$$\mathcal{M}_\alpha \models \{\alpha\} \cup \{\, \neg\alpha' \in \neg\Gamma \mid \alpha \not\models \alpha' \,\} \cup \neg\Delta.$$

Likewise, let $\mathcal{N}_\alpha = (\mathcal{K}^{\mathcal{N}_\alpha}, w^{\mathcal{N}_\alpha})$ be a model such that

$$\mathcal{N}_\alpha \models \{\alpha \wedge \lambda(\alpha)\} \cup \big\{\neg(\alpha' \wedge \lambda(\alpha')) \mid \alpha' \in \Gamma \setminus \{\alpha\} \text{ and } \alpha \models \alpha'\big\}.$$

We show that $\mathcal{N}_\alpha$ exists. Due to multiplicativity, consider a single $\alpha' \neq \alpha$ such that $\alpha \models \alpha'$. Γ is partially ordered, so $\alpha' \not\models \alpha$, and by **3.** $(\alpha \wedge \lambda(\alpha)) \not\models (\alpha' \wedge \lambda(\alpha'))$. For this reason, $(\alpha \wedge \lambda(\alpha)) \wedge \neg(\alpha' \wedge \lambda(\alpha'))$ is satisfiable.

$\mathcal{K}$ is now the disjoint union of all $\mathcal{M}_\alpha$,$\mathcal{N}_\alpha$ and an additional world r_α for each $\alpha \in \Gamma^*$, which possesses edges to the roots of $\mathcal{M}_\alpha$ and $\mathcal{N}_\alpha$ (see Figure 2). Finally, let $T := \{\, r_\alpha \mid \alpha \in \Gamma^* \,\}$. It is straightforward that the image of each $\{r_\alpha\} \subseteq T$ satisfies the disjunct $\alpha \wedge \mathsf{E}\lambda(\alpha)$ of φ, so $T \models \mathsf{AS}\Box\varphi$.

Now suppose that $\approx$ is an equivalence relation with index less than $|\Gamma^*|$. Then $r_\alpha \approx r_{\alpha'}$ for some distinct $\alpha, \alpha' \in \Gamma^*$, i.e., the equivalence relation inevitably merges two worlds in T. Let $\approx'$ be the refinement of $\approx$ that merges nothing else. Suppose for the sake of contradiction that $\approx'$ preserves $\mathsf{AS}\Box\varphi$ on $(\mathcal{K}, T)$.

By definition of $\mathcal{M}_\alpha$, it holds $[r_\alpha]_{\approx'} \models \Diamond\neg\alpha''$, unless both $\alpha \models \alpha''$ and $\alpha' \models \alpha''$. Consequently, for $\{[r_\alpha]_{\approx'}\}$ to satisfy $\Box\varphi$, its image must necessarily satisfy a disjunct $\alpha'' \wedge \mathsf{E}\lambda(\alpha'')$ of φ with $\alpha \models \alpha''$ and $\alpha' \models \alpha''$.

Suppose such α'' exists. As Γ^* is an antichain, α, α' and α'' must be pairwise distinct. But then $(\alpha'' \wedge \lambda(\alpha''))$ is already false in the roots of $\mathcal{M}_\alpha$, $\mathcal{M}_{\alpha'}$, $\mathcal{N}_\alpha$, and $\mathcal{N}_{\alpha'}$. Since $\{[r_\alpha]_{\approx'}\} \models \Box{\sim}\alpha''$ or $\{[r_\alpha]_{\approx'}\} \models \Box\neg\lambda(\alpha'')$, α'' cannot exist. For this reason, ultimately $\{[r_\alpha]_{\approx'}\} \not\models \Box\varphi$ holds and $\approx$ is not strongly $\mathsf{AS}\Box\varphi$-preserving. ◀

Of course we still have to ask whether suitable DNFs with large antichains actually exist. In Subsection 5.1, we introduce a candidate, namely the *naive expansion* of $\mathcal{MTL}$

formulas. In Subsection 5.2, we then investigate the naive expansion of formulas of the form $\underbrace{\sim\Diamond\sim\Diamond\cdots}_{n \text{ alternations}}\varphi$, and prove that its width grows faster than any elementary function in n.

5.1 Naive DNF expansions

Suppose that $\varphi = (\Gamma, \Delta, \lambda)$ is a DNF. If $\sim\varphi$ should hold in a team, then for each $\alpha \in \Gamma$, either $\mathsf{E}\neg\alpha$ holds, or $\neg\beta$ is true for some $\beta \in \lambda(\alpha)$. Depending on λ, there are many possible ways to falsify φ. We represent each one by a *choice function* $\mathfrak{f}_\varphi$. By convention, $\mathfrak{f}_\varphi(\alpha)$ being undefined shall mean that $\mathsf{E}\neg\alpha$ holds, and otherwise $\neg\mathfrak{f}_\varphi(\alpha)$ is true. The set of all choice functions with respect to φ is then:

$$F_\varphi = \{\, \mathfrak{f} : \Gamma' \to \Delta \mid \Gamma' \subseteq \Gamma \text{ and } \forall \alpha \in \Gamma' : \mathfrak{f}(\alpha) \in \lambda(\alpha) \,\}.$$

To now express $\sim\varphi$ as a DNF, every possible choice function $\mathfrak{f}$ must be considered. The DNF $(\Gamma, \Delta, \lambda)^\sim$, logically equivalent to $\sim(\Gamma, \Delta, \lambda)$, is defined as $(\Gamma', \Delta', \lambda')$ with

$$\Gamma' := \left\{ \bigwedge_{\alpha \in \mathsf{dom}\,\mathfrak{f}} \neg\mathfrak{f}(\alpha) \;\middle|\; \mathfrak{f} \in F_\varphi \right\}$$
$$\Delta' := \neg\Gamma,$$

and $\lambda'\left(\bigwedge_{\alpha\in\mathsf{dom}\,\mathfrak{f}} \neg\mathfrak{f}(\alpha)\right) := \neg(\Gamma \setminus \mathsf{dom}\,\mathfrak{f})$.

For $\Diamond$, recall that $\Box, \Diamond$ and $\vee$ all distribute over Boolean disjunction $\veebar$. But unlike the $\Box$ operator, $\Diamond$ and $\vee$ do not distribute over conjunction. Here, we merely have a "pseudo-distributive law" for $\Diamond$ (and a similar one for $\vee$),

$$\Diamond\left(\alpha \wedge \bigwedge_{i=1}^{n} \mathsf{E}\beta_i\right) \equiv \Diamond\alpha \wedge \bigwedge_{i=1}^{n} \mathsf{E}\Diamond(\alpha \wedge \beta_i),$$

where $\alpha, \beta \in \mathcal{ML}$ [14]. Then the DNF $(\Gamma, \Delta, \lambda)^\Diamond := (\Diamond\Gamma, \Delta', \lambda')$, where $\Delta' := \{\Diamond(\alpha \wedge \beta) \mid \alpha \in \Gamma, \beta \in \lambda(\alpha)\}$ and $\lambda'(\Diamond\alpha) := \{\Diamond(\alpha \wedge \beta) \mid \beta \in \lambda(\alpha)\}$, is equivalent to $\Diamond(\Gamma, \Delta, \lambda)$.

All in all, the combination of the above steps yields the following DNF $(\Gamma, \Delta, \lambda)^{\Diamond\sim} = (\Gamma', \Delta', \lambda')$ equivalent to $\sim\Diamond(\Gamma, \Delta, \lambda)$. We gather the "positive" literals of $\mathfrak{f} \in F_\varphi$ in the conjunction $\ulcorner\mathfrak{f}\urcorner := \bigwedge_{\alpha\in\mathsf{dom}\,\mathfrak{f}} \neg(\alpha \wedge \mathfrak{f}(\alpha))$, and simply define:

$$\Gamma' := \{\, \Box\ulcorner\mathfrak{f}\urcorner \mid \mathfrak{f} \in F_\varphi \,\}$$
$$\Delta' := \neg\Diamond\Gamma$$
$$\lambda'(\Box\ulcorner\mathfrak{f}\urcorner) := \neg\Diamond(\Gamma \setminus \mathsf{dom}\,\mathfrak{f}).$$

5.2 Existence of large width DNFs

In the naive expansion $(\Gamma, \Delta, \lambda)$ of a formula, many literals in Γ or Δ may happen to be redundant. To ensure that Γ contains a large antichain, and even is partially ordered under $\models$, we have to carefully "carve out" superfluous disjuncts. As a first step, we formalize a sufficient notion of redundancy of disjuncts in a DNF.

If f is a function and $X \subseteq \mathsf{dom}\,f$, let $f{\restriction_X}$ be the restriction of f to the domain X.

▶ **Definition 27.** Let $\varphi = (\Gamma, \Delta, \lambda)$ be a DNF. A *kernel* of φ is a DNF $\varphi' = (\Gamma', \Delta', \lambda')$ such that

1. $\Gamma' \subseteq \Gamma$, $\lambda' = \lambda{\restriction_{\Gamma'}}$, and $\Delta' = \bigcup_{\alpha\in\Gamma'} \lambda'(\alpha)$
2. for every $\alpha \in \Gamma$ there exists $\alpha' \in \Gamma'$ such that $\alpha \vDash \alpha'$ and $\lambda'(\alpha') \subseteq \lambda(\alpha)^{\uparrow}$.

The condition $\lambda'(\alpha') \subseteq \lambda(\alpha)^{\uparrow}$ can also be formulated as: for all $\beta' \in \lambda'(\alpha')$ it holds $\beta \vDash \beta'$ for some $\beta \in \lambda(\alpha)$.

▶ **Lemma 28.** *If φ' is a kernel of φ, then $\varphi \equiv \varphi'$.*

Proof. Since φ includes all disjuncts of φ', clearly $\varphi' \vDash \varphi$. Conversely, assume that a model $\mathcal{M}$ satisfies a disjunct $\alpha \wedge \mathsf{E}\beta_1 \wedge \dots \wedge \mathsf{E}\beta_n$ of φ. Then there is a disjunct of φ' of the form $\alpha' \wedge \mathsf{E}\beta'_1, \wedge \dots \wedge \mathsf{E}\beta'_m$, where $\alpha \vDash \alpha'$ and for every $i \in [m]$ there exists $j \in [n]$ such that $\beta_j \vDash \beta'_i$. Consequently, $\mathcal{M}$ satisfies $\alpha' \wedge \mathsf{E}\beta'_1, \wedge \dots \wedge \mathsf{E}\beta'_m$ and therefore φ'. ◀

In the next lemma, we ensure that the Hanoi property, Definition 24, is preserved under application of $\sim\Diamond$.

▶ **Lemma 29.** *If a DNF $(\Gamma, \Delta, \lambda)$ has the Hanoi property, then also any kernel of $(\Gamma, \Delta, \lambda)^{\Diamond\sim}$ has it.*

Proof. Let $(\Gamma', \Delta', \lambda') := (\Gamma, \Delta, \lambda)^{\Diamond\sim}$. Proving the lemma for $(\Gamma', \Delta', \lambda')$ also proves it for every kernel.

Proof of condition 1. Since $\Gamma \cup \neg\Delta$ is satisfiable, clearly $\Diamond(\Gamma \cup \neg\Delta) \cup \Box(\Gamma \cup \neg\Delta)$ is satisfiable. The claim follows since $\Box\neg\Delta \vDash \Gamma'$ and $\Diamond\Gamma \equiv \neg\neg\Diamond\Gamma = \neg\Delta'$.

Proof of condition 2. Suppose $\Gamma'_1 \cup \neg\Gamma'_2$ is satisfiable for $\Gamma'_1, \Gamma'_2 \subseteq \Gamma'$. Note that $\Gamma' \cup \Delta'$ is multiplicative, since every formula in it is equivalent to an element of $\{\Box\varphi \mid \varphi \in \mathcal{ML}\}$ (cf. page 10). Since $\Gamma' \cup \neg\Delta'$ is satisfiable by condition **1.**, also $\Gamma'_1 \cup \neg\Gamma'_2 \cup \neg\Delta'$ is satisfiable.

Proof of condition 3. Let $\Delta'_1 \cup \neg\Delta'_2$ be satisfied by a model $\mathcal{M} = (\mathcal{K}, w)$. We augment $\mathcal{M}$ to additionally satisfy Γ'. Every formula in $\neg\Delta'_2$ is of the form $\neg\neg\Diamond\alpha$ for some $\alpha \in \Gamma$. The root w has a successor v_α for every such α such that $v_\alpha \vDash \{\alpha\} \cup \{\neg\alpha' \mid \neg\Diamond\alpha' \in \Delta'_1\}$. As condition **2.** holds for (Γ, Δ), we can for each such v_α add a new successor that is the root of a model of $\{\alpha\} \cup \{\neg\alpha' \mid \neg\Diamond\alpha' \in \Delta'_1\} \cup \neg\Delta$. If now all old successors of w are removed, the resulting structure satisfies $\Delta'_1 \cup \neg\Delta'_2 \cup \Gamma'$. ◀

In the rest of the section, we aim at applying Theorem 26 to a DNF $\varphi = (\Gamma, \Delta, \lambda)$ that is constructed to have a large width $w(\Gamma)$. The idea is to introduce alternations between positive and negative occurrences of modal operators, as stated at the beginning of this section. A lower bound of $w(\Gamma)$ is then proven by induction on the number of alternations. Unfortunately, the conditions **1.**–**3.** of Theorem 26 are not sufficient, since φ meeting them does not imply $\varphi^{\Diamond\sim}$ meeting them. Instead, we consider the following extended conditions which are used as an invariant during the induction step.

1. φ has the Hanoi property and $\Gamma \cup \Delta$ is multiplicative,
2. $\vDash$ is a partial ordering on both Γ and Δ,
3. **(a)** $\forall\alpha \in \Gamma : |\lambda(\alpha)| = 1$, and if $\lambda(\alpha) = \{\beta\}, \lambda(\alpha') = \{\beta'\}$, then $\alpha \vDash \alpha' \;\Leftrightarrow\; (\alpha' \wedge \beta') \vDash (\alpha \wedge \beta)$,
   (b) $\lambda : \Gamma \to \mathcal{U}(\Delta)$ is bijective and $\alpha \vDash \alpha' \;\Leftrightarrow\; \lambda(\alpha) \subseteq \lambda(\alpha')$,
4. **(a)** if $\alpha, \alpha' \in \Gamma$ are incomparable and $\beta \in \lambda(\alpha)$, then $\alpha \wedge \beta \nvDash \alpha'$,
   (b) if $\lambda(\alpha) \setminus \lambda(\alpha')$ contains an element incomparable to $\beta \in \Delta$, then $\alpha \wedge \beta \nvDash \alpha'$.

Moreover, if φ is a DNF with the properties **3.(a)** and **4.(a)**, then $\varphi^{\Diamond\sim}$ does not necessarily have a kernel which satisfies **3.(a)** and **4.(a)** again. Instead, the iterated expansion of a $\sim\Diamond$ prefix alternates between DNFs satisfying either **3.(a)–4.(a)** or **3.(b)–4.(b)**. This is made

explicit in the following two technical lemmas. The proofs can be found in the appendix. Note that the lemmas work asymmetrically: the width $w(\Gamma)$ stays the same in the first lemma, but increases by a factorial in the second lemma.

▶ **Lemma 30.** *If a DNF $\varphi = (\Gamma, \Delta, \lambda)$ satisfies **1.**, **2.**, **3.(a)** and **4.(a)**, then $\varphi^{\Diamond\sim}$ has a kernel $(\Gamma', \Delta', \lambda')$ that satisfies **1.**, **2.**, **3.(b)** and **4.(b)** such that $w(\Delta') = w(\Gamma)$.*

▶ **Lemma 31.** *If a DNF $\varphi = (\Gamma, \Delta, \lambda)$ satisfies **1.**, **2.**, **3.(b)** and **4.(b)**, then $\varphi^{\Diamond\sim}$ has a kernel $(\Gamma', \Delta', \lambda')$ that satisfies **1.**, **2.**, **3.(a)** and **4.(a)** such that $w(\Gamma') \geq w(\Delta)!$.*

Applying the two lemmas in an inductive manner yields a width that grows roughly as fast as the iterated factorial. We write the n-times iterated factorial of n as:

$$n!^{(n)} := n\underbrace{!\ldots!}_{n \text{ times}}$$

The iteration count is essentially half the *negation depth* $\mathrm{nd}(\varphi)$, the maximal nesting depth of $\sim$ in φ. It is defined similarly as the *modal depth* $\mathrm{md}(\varphi)$, which is the maximal nesting depth of modalities [3].

It is not difficult to choose an initial DNF φ that satisfies **1.**, **2.**, **3.(a)** and **4.(a)** A simple example is $\varphi := \bigvee_{i=1}^{n}(p_i \wedge \mathsf{E}q_i)$, where $p_1, q_1, \ldots \in \mathcal{PS}$ are distinct. By applying Lemma 30 and 31 repeatedly, we obtain a DNF φ'. φ' satisfies **1.**, **2.**, **3.(a)** and **4.(a)**, and is by Lemma 28 equivalent to $(\sim\Diamond)^{2n}\varphi$, but its positive literal set has width at least $n!^{(n)}$.

Clearly an equivalence relation is φ-preserving if and only if it is φ'-preserving. A consequence of this fact and of Theorem 26 is the following main result of this section.

▶ **Theorem 32.** *There are $\mathcal{MTL}$ formulas $(\varphi_n)_{n\in\mathbb{N}}$ of length $\mathcal{O}(n)$, modal depth $2n + \mathcal{O}(1)$ and negation depth $2n + \mathcal{O}(1)$, and models $(\mathcal{K}, T)$ of φ_n, such that every equivalence relation that strongly preserves φ_n on $(\mathcal{K}, T)$ has index at least $n!^{(n)}$.*

Recall that $\mathcal{MTL}_{\mathsf{mon}}$ has, according to Theorem 18, strongly φ-preserving filtration with index exponential in $|\varphi|$. Therefore we conclude this section with a non-elementary succinctness gap between the equally expressive logics $\mathcal{MTL}$ and $\mathcal{MTL}_{\mathsf{mon}}$.

▶ **Corollary 33.** *There are $\mathcal{MTL}$ formulas $(\varphi_n)_{n\in\mathbb{N}}$ of length $\mathcal{O}(n)$, modal depth $2n + \mathcal{O}(1)$ and negation depth $2n + \mathcal{O}(1)$ for which any equivalent $\mathcal{MTL}_{mon}$ formula has length at least $n!^{(n)}$.*

6 Conclusion

In this paper, the filtration technique was shown to be insufficient as a tool applied to modal logic considered under team semantics and supplied with Boolean negation.

On the one hand, filtration continues to work in an almost straightforward way for the extension $\mathcal{S}(\mathcal{ML})$ of $\mathcal{ML}$ – that is, the closure of $\mathcal{ML}$ under splitting $\vee$ and Boolean operators $\sim, \wedge$. Theorem 9, the exponential model property for $\mathcal{S}(\mathcal{ML})$, is not particularly surprising: for a given $\Gamma \subseteq \mathcal{ML}$, let Γ^* be the closure of Γ under $\neg$ and $\wedge$. Then every $\mathcal{S}(\Gamma)$ formula is equivalent to a $\mathcal{B}(\Gamma^*)$ formula [14].[5] As filtration upper bounds are clearly "closed under Boolean connectives", including all $\wedge$, $\neg$ and $\sim$, the strong upper bound in Section 3 naturally emerges.

[5] The translation from the first fragment to the latter plays a crucial role in the completeness of a recent axiomatic system for $\mathcal{MTL}$ [14].

On the other hand, the introduction of team-wide modalities prevents filtration very thoroughly and effectively, as Section 5 shows. The proof of the non-elementary filtration lower bound employs a formula $\mathsf{AS}\Box(\sim\Diamond)^{2n}\varphi$ where φ is a disjunctive normal form (using $\sim$ and $\wedge$) of only propositional variables. Besides $\Diamond$ operators alternating between positive and negative occurrences, for technical reasons that formula contains one additional $\Box$ and, contained in the definition of the AS operator, two instances of $\vee$.

The main conclusion that can be drawn from these results is the following. Filtration, being defined as a simple quotient construction merging individual worlds, preserves modal formulas that can be evaluated on points, but it cannot capture the interdependencies between different points in a Kripke structure that are addressed by the team-wide modalities.

Another subtle point is the distinction between an equivalence relation preserving the truth value of a formula φ on a pair $(\mathcal{K}, T)$, or even in all teams throughout the whole structure $\mathcal{K}$. Intuitively, the first variant corresponds to choosing a filtration after knowing the team T in which φ must be preserved, while the latter one corresponds to not knowing T beforehand. While $\mathcal{S}(\mathcal{ML})$ admits filtration in both senses, Section 3.1 shows that already the introduction of a single occurrence of a non-classical modality prohibits a filtration of exponential size unless T is fixed, i.e., known beforehand.

This more specialized approach is developed in Section 4 to a "weak filtration" for the fragment named $\mathcal{MTL}_{\mathsf{mon}}$. It has the exponential model property, at the cost of disallowing $\Diamond$ and $\Box$ to occur negatively. While this modal fragment appears rather artificial to allow filtration, it potentially permits a variant of the standard translation (see, e.g., [3]) to the existential fragment of a variant of first-order or second-order logic. It is well-known that the satisfiability problem for the analogous existential $\mathsf{SO}(\exists)$ fragment of second-order logic is not harder than for first-order logic FO.

The paper aims at classifying the actual complexity of Modal Team Logic $\mathcal{MTL}$, in the sense of both computational and model-theoretical complexity. From that perspective, the paper is clearly a negative result. Nevertheless, the used proof methods for the failure of filtration have useful side effects: namely they imply several succinctness lower bounds, even indicating a strict succinctness hierarchy of fragments inside $\mathcal{MTL}$. The detailed structure of that hierarchy is a potential target of future research, as well as closing the gap of known unconditional upper and lower bounds for model size.

Acknowledgements. The author would like to express his gratitude to the anonymous referees for their numerous helpful comments.

References

1 Samson Abramsky and Jouko Väänänen. From IF to BI. *Synthese*, 167(2):207–230, 2009. doi:10.1007/s11229-008-9415-6.

2 Ernest Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science (Vol. B)*, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990. URL: http://dl.acm.org/citation.cfm?id=114891.114907.

3 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal logic*. Cambridge University Press, New York, NY, USA, 2001.

4 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981. doi:10.1145/322234.322243.

5 Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order*. Cambridge university press, 2002.

6 Pietro Galliani. Inclusion and exclusion dependencies in team semantics – On some logics of imperfect information. *Annals of Pure and Applied Logic*, 163(1):68–84, January 2012. doi:10.1016/j.apal.2011.08.005.

7 Erich Grädel and Jouko Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013.

8 Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the Decision Problem for Two-Variable First-Order Logic. *Bulletin of Symbolic Logic*, 3(01):53–69, March 1997. doi:10.2307/421196.

9 Miika Hannula, Juha Kontinen, Martin Lück, and Jonni Virtema. On Quantified Propositional Logics and the Exponential Time Hierarchy. In *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016.*, pages 198–212, 2016. doi:10.4204/EPTCS.226.14.

10 Miika Hannula, Juha Kontinen, Jonni Virtema, and Heribert Vollmer. Complexity of Propositional Independence and Inclusion Logic. In Giuseppe F Italiano, Giovanni Pighizzini, and Donald T. Sannella, editors, *Mathematical Foundations of Computer Science 2015*, volume 9234, pages 269–280. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. doi:10.1007/978-3-662-48057-1_21.

11 Jaakko Hintikka and Gabriel Sandu. Informational Independence as a Semantical Phenomenon. In *Studies in Logic and the Foundations of Mathematics*, volume 126, pages 571–589. Elsevier, 1989.

12 Wilfrid Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of IGPL*, 5(4):539–563, July 1997. doi:10.1093/jigpal/5.4.539.

13 Juha Kontinen, Julian-Steffen Müller, Henning Schnoor, and Heribert Vollmer. A Van Benthem Theorem for Modal Team Semantics. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, pages 277–291, 2015. doi:10.4230/LIPIcs.CSL.2015.277.

14 Martin Lück. Axiomatizations for Propositional and Modal Team Logic. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:18, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2016.33.

15 Julian-Steffen Müller. *Satisfiability and Model Checking in team based logics*. PhD thesis, University of Hanover, 2014. URL: http://d-nb.info/1054741921.

16 Jouko Väänänen. Modal dependence logic. *New perspectives on games and interaction*, 4:237–254, 2008.

17 Jouko Väänänen. *Dependence logic: a new approach to independence friendly logic*. Number 70 in London Mathematical Society student texts. Cambridge University Press, Cambridge ; New York, 2007.

A Technical Appendix: Proofs for Section 5.2

A.1 Proofs of lemmas

▶ **Lemma 30.** *If a DNF $\varphi = (\Gamma, \Delta, \lambda)$ satisfies **1., 2., 3.(a)** and **4.(a)**, then $\varphi^{\Diamond\sim}$ has a kernel $(\Gamma', \Delta', \lambda')$ that satisfies **1., 2., 3.(b)** and **4.(b)** such that $w(\Delta') = w(\Gamma)$.*

Proof. We form the kernel $(\Gamma', \Delta', \lambda')$ of $(\Gamma'', \Delta'', \lambda'') := \varphi^{\Diamond\sim}$ using exactly the choice functions with upward closed domain, i.e., $F' := \{\, \mathfrak{f} \in F_\varphi \mid \mathsf{dom}\,\mathfrak{f} \in \mathcal{U}(\Gamma) \,\}$.

To meet the conditions of a kernel (cf. Definition 27), for every $\mathfrak{f} \in F_\varphi$ there has to exist $\mathfrak{f}' \in F'$ such that $\Box\ulcorner\mathfrak{f}\urcorner \models \Box\ulcorner\mathfrak{f}'\urcorner$ and $\lambda''(\Box\ulcorner\mathfrak{f}'\urcorner) \subseteq \lambda''(\Box\ulcorner\mathfrak{f}\urcorner)^{\uparrow}$. Obtain such $\mathfrak{f}'$ by restricting

$\mathfrak{f}$ to the domain $\Gamma \setminus (\Gamma \setminus \mathsf{dom}\,\mathfrak{f})^{\downarrow}$. Then $\mathsf{dom}\,\mathfrak{f}'$ is upward closed. Also, trivially $\ulcorner\mathfrak{f}\urcorner \vDash \ulcorner\mathfrak{f}'\urcorner$. Furthermore,

$$\lambda''(\Box\ulcorner\mathfrak{f}'\urcorner) = \neg\Diamond\big((\Gamma \setminus \mathsf{dom}\,\mathfrak{f})^{\downarrow}\big) = (\neg\Diamond(\Gamma \setminus \mathsf{dom}\,\mathfrak{f}))^{\uparrow} = \lambda''(\Box\ulcorner\mathfrak{f}\urcorner)^{\uparrow}.$$

Consequently, $(\Gamma', \Delta', \lambda')$ with $\Gamma' := \{\Box\ulcorner\mathfrak{f}\urcorner \mid \mathfrak{f} \in F'\}$, and Δ', λ' defined accordingly, is a kernel. It is multiplicative and Lemma 29 applies, so condition **1.** is satisfied. Δ' is partially ordered and $w(\Delta') = w(\Gamma)$, since $\Delta' = \neg\Diamond\Gamma$. That Γ' is partially ordered follows from the proof of **3.(b)**, as $\subseteq$ is a partial order.

To prove the lemma, it remains to show **3.(b)** and **4.(b)**.

Proof of 3.(b) For every $\mathfrak{f} \in F'$, it is easy to verify that $\lambda'(\Box\ulcorner\mathfrak{f}\urcorner)$ is an upward closed subset of Δ', so $\lambda' : \Gamma' \to \mathcal{U}(\Delta')$. We prove only $\mathsf{dom}\,\mathfrak{f}' \subseteq \mathsf{dom}\,\mathfrak{f} \Leftrightarrow \ulcorner\mathfrak{f}\urcorner \vDash \ulcorner\mathfrak{f}'\urcorner$ for all $\mathfrak{f}, \mathfrak{f}' \in F'$, since then

$$\begin{aligned} & \Box\ulcorner\mathfrak{f}\urcorner \vDash \Box\ulcorner\mathfrak{f}'\urcorner \\ \Leftrightarrow \quad & \ulcorner\mathfrak{f}\urcorner \vDash \ulcorner\mathfrak{f}'\urcorner \\ \Leftrightarrow \quad & \mathsf{dom}\,\mathfrak{f}' \subseteq \mathsf{dom}\,\mathfrak{f} \\ \Leftrightarrow\ & \lambda'(\Box\ulcorner\mathfrak{f}\urcorner) \subseteq \lambda'(\Box\ulcorner\mathfrak{f}'\urcorner) \quad \text{(by definition of } \lambda'\text{).} \end{aligned}$$

Recall that $\lambda'(\alpha)$ is a singleton, and consequently $\mathfrak{f}(\alpha) = \mathfrak{f}'(\alpha)$ for every $\alpha \in \mathsf{dom}\,\mathfrak{f} \cap \mathsf{dom}\,\mathfrak{f}'$. For this reason, $\mathsf{dom}\,\mathfrak{f}' \subseteq \mathsf{dom}\,\mathfrak{f}$ implies $\ulcorner\mathfrak{f}\urcorner \vDash \ulcorner\mathfrak{f}'\urcorner$. For the other direction, suppose $\mathsf{dom}\,\mathfrak{f}' \not\subseteq \mathsf{dom}\,\mathfrak{f}$, i.e., $\mathfrak{f}'(\alpha) = \beta$ for some $\alpha \in \mathsf{dom}\,\mathfrak{f}' \setminus \mathsf{dom}\,\mathfrak{f}$. We prove that

$$\delta := (\alpha \wedge \beta) \wedge \bigwedge_{\alpha' \in \mathsf{dom}\,\mathfrak{f}} \neg(\alpha' \wedge \mathfrak{f}(\alpha'))$$

and therefore $\neg\ulcorner\mathfrak{f}'\urcorner \wedge \ulcorner\mathfrak{f}\urcorner$ is satisfiable. Due to multiplicativity, we can simply ascertain the consistency of $(\alpha \wedge \beta)$ with all conjuncts $\neg(\alpha' \wedge \mathfrak{f}(\alpha'))$ of $\mathfrak{f}$. If $\alpha' \in \mathsf{dom}\,\mathfrak{f}$, then $\alpha' \not\vDash \alpha$, as $\mathsf{dom}\,\mathfrak{f}$ is upward closed. Then **3.(a)** implies that $(\alpha \wedge \beta) \wedge \neg(\alpha' \wedge \mathfrak{f}(\alpha'))$ is satisfiable. λ' is injective: if $\lambda'(\Box\ulcorner\mathfrak{f}\urcorner) = \lambda'(\Box\ulcorner\mathfrak{f}'\urcorner)$, then $\mathsf{dom}\,\mathfrak{f}' = \mathsf{dom}\,\mathfrak{f}$, and consequently $\mathfrak{f}' = \mathfrak{f}$ by the same argument as stated above. By cardinality constraints, λ' is then surjective, as $|\Gamma'| = |\mathcal{U}(\Gamma)| = |\mathcal{L}(\Gamma)| = |\mathcal{U}(\Delta')|$. It follows that λ' is a bijection.

Proof of 4.(b) Suppose that $\lambda'(\Box\ulcorner\mathfrak{f}\urcorner) \setminus \lambda'(\Box\ulcorner\mathfrak{f}'\urcorner)$ contains an element $\neg\Diamond\alpha'$ incomparable to $\neg\Diamond\alpha \in \Delta'$. First note that α and α' must be incomparable as well, and furthermore, $\alpha' \in \mathsf{dom}\,\mathfrak{f}' \setminus \mathsf{dom}\,\mathfrak{f}$ by definition of λ'. Similar as before, we show that

$$\delta' := (\alpha' \wedge \mathfrak{f}'(\alpha')) \wedge \neg\alpha \wedge \bigwedge_{\alpha'' \in \mathsf{dom}\,\mathfrak{f}} \neg(\alpha'' \wedge \mathfrak{f}(\alpha''))$$

is satisfiable, implying $\ulcorner\mathfrak{f}\urcorner \wedge \neg\alpha \not\vDash \ulcorner\mathfrak{f}'\urcorner$ and consequently **4.(b)**, as $\Box\ulcorner\mathfrak{f}\urcorner \wedge \Box\neg\alpha \not\vDash \Box\ulcorner\mathfrak{f}'\urcorner$. Finally, the satisfiability of $(\alpha' \wedge \mathfrak{f}'(\alpha')) \wedge \neg\alpha$ follows from **4.(a)**; while the satisfiability of $(\alpha' \wedge \mathfrak{f}'(\alpha')) \wedge \neg(\alpha'' \wedge \mathfrak{f}(\alpha''))$ follows again, since $\alpha' \notin \mathsf{dom}\,\mathfrak{f}$ implies $\alpha'' \not\vDash \alpha'$, from **3.(a)**

◀

Next we prove the converse lemma, in which the width increases by a factorial. For the sake of clarity, several auxiliary claims again are proven in the appendix after the lemma itself.

▶ **Lemma 31.** *If a DNF $\varphi = (\Gamma, \Delta, \lambda)$ satisfies **1.**, **2.**, **3.(b)** and **4b.**, then $\varphi^{\Diamond\sim}$ has a kernel $(\Gamma', \Delta', \lambda')$ that satisfies **1.**, **2.**, **3a.** and **4a.** such that $w(\Gamma') \geq w(\Delta)!$.*

Proof. As in Lemma 30, we construct a kernel $(\Gamma', \Delta', \lambda')$ of $(\Gamma'', \Delta'', \lambda'') := \varphi^{\Diamond\sim}$. By the same argument as there, w.l.o.g. all $\mathfrak{f} \in F_\varphi$ have an upward closed domain.

We introduce a number of definitions.

Due to **3.(b)**, λ actually is an order isomorphism between Γ and the lattice $\mathcal{U}(\Delta)$ (cf. [5]). For any $\widehat{\Delta} \subseteq \Delta$, the lattice $\mathcal{U}(\Delta)$ furthermore contains a sublattice $L_{\widehat{\Delta}}$ of all upward closed subsets of Δ which are disjoint from $\widehat{\Delta}$. Let $\alpha_{\widehat{\Delta}} := \lambda^{-1}\left(\max L_{\widehat{\Delta}}\right)$.

Let $S_{\mathrm{sub}}(\Delta)$ be the set of all permutations σ of subsets of Δ. If $\sigma = (\beta_1, \ldots, \beta_k) \in S_{\mathrm{sub}}(\Delta)$, then σ_j is the prefix $(\beta_i)_{i\in[j]}$ of σ for all $0 \leq j \leq k$. The set underlying σ is $\Delta_\sigma := \{\beta_i\}_{i\in[k]}$, and we also write α_σ instead of α_{Δ_σ}.

A permutation $\sigma = (\beta_1, \ldots, \beta_k)$ is *lazy* if β_j is a minimal element in $\lambda(\alpha_{\sigma_{j-1}})$ (which is not necessarily unique) for all $j \in [k]$. It is a *linear extension* of Δ_σ if, for all $i, j \in [k]$, $\beta_i \vDash \beta_j$ implies $i \leq j$.

▶ **Claim** (a). *$\sigma \in S_{\mathrm{sub}}(\Delta)$ is lazy if and only if it is a linear extension of some $\widehat{\Delta} \in \mathcal{L}(\Delta)$.*

For every lazy permutation $\sigma = (\beta_1, \ldots, \beta_k) \in S_{\mathrm{sub}}(\Delta)$, define the function $\mathfrak{f}_\sigma$ as follows. Let its domain $\mathsf{dom}\,\mathfrak{f}_\sigma$ be $\{\alpha_\emptyset\}^\downarrow \setminus \{\alpha_\sigma\}^\downarrow = \Gamma \setminus \{\alpha_\sigma\}^\downarrow$. For all $j \in [k]$ and $\alpha \in \{\alpha_{\sigma_{j-1}}\}^\downarrow \setminus \{\alpha_{\sigma_j}\}^\downarrow$, let $\mathfrak{f}(\alpha) = \beta_j$. Note that $\mathsf{dom}\,\mathfrak{f}_\sigma$ is an upward closed set. The range $\mathsf{ran}\,\mathfrak{f}_\sigma$ equals Δ_σ, which is a downward closed set by Claim (a).

Using these definitions, the restriction of F_φ to

$$F' := \{\, \mathfrak{f}_\sigma \mid \sigma \in S_{\mathrm{sub}}(\Delta) \text{ is lazy} \,\}$$

permits to define the desired DNF $(\Gamma', \Delta', \lambda')$ using

$$\Gamma' := \{\, \Box\ulcorner \mathfrak{f}_\sigma \urcorner \mid \sigma \in S_{\mathrm{sub}}(\Delta) \text{ is lazy} \,\},$$
$$\Delta' := \{\, \neg\Diamond\alpha_\sigma \mid \sigma \in S_{\mathrm{sub}}(\Delta) \text{ is lazy} \,\},$$

and $\lambda'(\Box\ulcorner \mathfrak{f}_\sigma \urcorner) := \neg\Diamond\alpha_\sigma$.

▶ **Claim** (b). *$(\Gamma', \Delta', \lambda')$ is a kernel of $\varphi^{\Diamond\sim}$.*

It is straightforward that $(\Gamma', \Delta', \lambda')$ satisfies **1.** We proceed with the remaining properties stated in the lemma, using another auxiliary claim.

▶ **Claim** (c). *Let $\mathfrak{f}_\sigma, \mathfrak{f}_\tau \in F'$ be distinct.*

- *If σ is a prefix of τ, then $\ulcorner \mathfrak{f}_\tau \urcorner \vDash \ulcorner \mathfrak{f}_\sigma \urcorner$ and $\ulcorner \mathfrak{f}_\sigma \urcorner \nvDash \ulcorner \mathfrak{f}_\tau \urcorner$, and moreover $\ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma \vDash \ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau$ and $\ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau \nvDash \ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma$.*
- *If σ is not a prefix of τ and vice versa, then $\ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma \nvDash \ulcorner \mathfrak{f}_\tau \urcorner$ and $\ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau \nvDash \ulcorner \mathfrak{f}_\sigma \urcorner$.*

The kernel satisfies **2.**: Clearly $\Delta' \subseteq \neg\Diamond\Gamma$ is partially ordered, since Γ is. From the above claim it also follows that Γ' is partially ordered as well, since never both $\Box\ulcorner \mathfrak{f}_\sigma \urcorner \vDash \Box\ulcorner \mathfrak{f}_\tau \urcorner$ and $\Box\ulcorner \mathfrak{f}_\sigma \urcorner \vDash \Box\ulcorner \mathfrak{f}_\tau \urcorner$ hold simultaneously for $\sigma \neq \tau$. Moreover, in both cases **3.(a)** and **4.(a)** apply.

Finally, we determine $w(\Gamma')$. Let $\widehat{\Delta} \subseteq \Delta$ be an antichain with cardinality $n := w(\Delta)$. There are at least $n!$ distinct linear extensions of $(\widehat{\Delta})^\downarrow$, obtained by freely arranging the elements in $\widehat{\Delta}$, and by Claim (a) then at least $n!$ distinct lazy permutations of $(\widehat{\Delta})^\downarrow$. Since these permutations are of identical length, they are pairwise no prefix of each other. By Claim (c), then Γ' contains an antichain of size $n!$. ◀

A.2 Proofs of claims

▶ **Claim** (a). *$\sigma \in S_{\mathrm{sub}}(\Delta)$ is lazy if and only if is a linear extension of some $\widehat{\Delta} \in \mathcal{L}(\Delta)$.*

Proof. Let $\sigma = (\beta_1, \dots, \beta_k)$ be a linear extension of $\widehat{\Delta} \in \mathcal{L}(\Delta)$. We prove that β_j is a minimal element of $\lambda(\alpha_{\sigma_{j-1}})$ for all $j \in [k]$, so σ is lazy.

Observe that the elements $\alpha_{\sigma_0}, \alpha_{\sigma_1}, \dots, \alpha_{\sigma_k}$ form an descending chain in Γ whenever σ is lazy, that is, $\alpha_\sigma = \alpha_{\sigma_k} \vDash \alpha_{\sigma_{k-1}} \vDash \cdots \vDash \alpha_{\sigma_1} \vDash \alpha_{\sigma_0} = \alpha_\emptyset$.

First note that, for every $j \in [k]$, $\widehat{\Delta}_j := \{\beta_i \mid i \in [j]\}$ is downward closed as well. Conversely, the set $\Delta \setminus \widehat{\Delta}_{j-1}$ is disjoint from $\widehat{\Delta}_{j-1}$ and upward closed. β_j is an element of $\lambda(\alpha_{\sigma_{j-1}})$, since $\beta_j \in \Delta \setminus \widehat{\Delta}_{j-1}$ and $\lambda(\alpha_{\sigma_{j-1}})$ is the largest upper set disjoint from $\widehat{\Delta}_{j-1}$. β_j is indeed a minimal element: suppose that $\lambda(\alpha_{\sigma_{j-1}})$ contains an element below β_j. That element is in $\widehat{\Delta}$ by downward closure, so it equals some β_ℓ in σ. It holds $\ell < j$, since σ is a linear extension. Also, $\beta_\ell \notin \lambda(\alpha_{\sigma_\ell})$ by definition of α_{σ_ℓ}. But since $\lambda(\alpha_\emptyset) \supseteq \lambda(\alpha_{\sigma_1}) \supseteq \cdots \supseteq \lambda(\alpha_\sigma)$, β_ℓ is not in $\lambda(\alpha_{\sigma_{j-1}})$ either, contradiction. So σ is lazy.

For the other direction, assume that $\sigma = (\beta_1, \dots, \beta_k)$ is lazy. Define $\widehat{\Delta}_j$ as above. As a first step, we prove by induction on j that $\widehat{\Delta}_j$ is downward closed. As $j = 0$ is trivial, consider $j > 0$. Since every prefix of a lazy permutation is lazy, $\widehat{\Delta}_{j-1}$ is downward closed by induction hypothesis. Furthermore, β_j is a minimal element of $\lambda(\alpha_{\sigma_{j-1}})$ by assumption. $\lambda(\alpha_{\sigma_{j-1}})$ is the largest upward closed set disjoint to $\widehat{\Delta}_{j-1}$, so by downward closure of $\widehat{\Delta}_{j-1}$ it must equal $\Delta \setminus \widehat{\Delta}_{j-1}$. But then β_j is minimal in $\Delta \setminus \widehat{\Delta}_{j-1}$. As any element below β_j is in $\widehat{\Delta}_{j-1}$, the set $\widehat{\Delta}_{j-1} \cup \{\beta_j\} = \widehat{\Delta}_j$ is downward closed.

Finally, suppose σ is lazy, but not a linear extension of its underlying set $\widehat{\Delta}$. Then there are β_i, β_j such that $i < j$ and $\beta_j \vDash \beta_i$. It holds $\lambda(\alpha_{\sigma_i}) \supseteq \lambda(\alpha_{\sigma_j})$. But then $\beta_j \in \lambda(\alpha_{\sigma_i})$, which contradicts the fact that β_i is a minimal element of $\lambda(\alpha_{\sigma_i})$. ◀

▶ **Claim** (b). *$(\Gamma', \Delta', \lambda')$ is a kernel of $\varphi^{\Diamond\sim}$.*

Proof. Let $(\Gamma'', \Delta'', \lambda'') := \varphi^{\Diamond\sim}$. Say that $\mathfrak{f}$ *covers* $\mathfrak{f}'$ if $\ulcorner \mathfrak{f} \urcorner \vDash \ulcorner \mathfrak{f}' \urcorner$ and $\lambda'(\Box\ulcorner \mathfrak{f}' \urcorner) \subseteq \lambda'(\Box\ulcorner \mathfrak{f} \urcorner)^\uparrow$. If $\sigma = (\beta_1, \dots, \beta_n)$ is a permutation not containing the element β_{n+1}, then (σ, β_{n+1}) denotes the permutation $(\beta_1, \dots, \beta_n, \beta_{n+1})$.

We prove that if $\mathfrak{f}_\sigma$ is the restriction of a function $\mathfrak{f}$, then either $\mathfrak{f}_\sigma = \mathfrak{f}$, or there exists $\beta \in \Delta$ such that $\mathfrak{f}_{(\sigma,\beta)}$ is the restriction of some $\mathfrak{f}' \in F$ covering $\mathfrak{f}$. Since covering is transitive, and since $\mathfrak{f}_()$ is the restriction of every function, by induction on $|\sigma|$ every $\mathfrak{f} \in F_\varphi$ is covered by some $\mathfrak{f}_\sigma \in F'$, proving the claim.

For the proof, assume $\mathfrak{f}_\sigma \neq \mathfrak{f}$, i.e., $\mathsf{dom}\,\mathfrak{f}_\sigma \subsetneq \mathsf{dom}\,\mathfrak{f}$. Then $\alpha_\sigma \in \mathsf{dom}\,\mathfrak{f}$, since $\mathsf{dom}\,\mathfrak{f} = \Gamma \setminus \{\alpha_\sigma\}^\downarrow$ and $\mathsf{dom}\,\mathfrak{f}$ is upward closed. Let β^* be a minimal element of $\lambda(\alpha_\sigma)$ below or equal to $\mathfrak{f}(\alpha_\sigma)$. Define

$$\mathfrak{f}'(\alpha) = \begin{cases} \beta^* \text{ if } \alpha \in \{\alpha_\sigma\}^\downarrow \setminus \{\alpha_{(\sigma,\beta')}\}^\downarrow, \\ \mathfrak{f}(\alpha) \text{ otherwise.} \end{cases}$$

Then $\mathfrak{f}_{(\sigma,\beta^*)}$ is a restriction of $\mathfrak{f}'$. That $\mathfrak{f}'$ covers $\mathfrak{f}$ can be seen as follows: Clearly $\lambda''(\Box\ulcorner \mathfrak{f}' \urcorner) \subseteq \lambda''(\Box\ulcorner \mathfrak{f} \urcorner)$, since $\mathsf{dom}\,\mathfrak{f}' \supseteq \mathsf{dom}\,\mathfrak{f}$. To show $\ulcorner \mathfrak{f} \urcorner \vDash \ulcorner \mathfrak{f}' \urcorner$, let $\neg(\alpha \wedge \beta)$ be a conjunct of $\ulcorner \mathfrak{f}' \urcorner$ not occurring in $\ulcorner \mathfrak{f} \urcorner$. Then $\alpha \in \{\alpha_\sigma\}^\downarrow$ and $\beta = \beta^*$, so $(\alpha \wedge \beta) \vDash (\alpha_\sigma \wedge \mathfrak{f}(\alpha_\sigma))$. It follows that $\ulcorner \mathfrak{f} \urcorner$ has a conjunct $\neg(\alpha_\sigma \wedge \mathfrak{f}(\alpha_\sigma))$ that implies $\neg(\alpha \wedge \beta)$. ◀

▶ **Claim** (c). *Let $\mathfrak{f}_\sigma, \mathfrak{f}_\tau \in F'$ be distinct.*

- *If σ is a prefix of τ, then $\ulcorner \mathfrak{f}_\tau \urcorner \vDash \ulcorner \mathfrak{f}_\sigma \urcorner$ and $\ulcorner \mathfrak{f}_\sigma \urcorner \nvDash \ulcorner \mathfrak{f}_\tau \urcorner$, and moreover $\ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma \vDash \ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau$ and $\ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau \nvDash \ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma$.*
- *If σ is not a prefix of τ and vice versa, then $\ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma \nvDash \ulcorner \mathfrak{f}_\tau \urcorner$ and $\ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau \nvDash \ulcorner \mathfrak{f}_\sigma \urcorner$.*

Proof of Claim (c).

(i) σ is a prefix of τ. Then $\mathsf{dom}\, \mathfrak{f}_\sigma \subseteq \mathsf{dom}\, \mathfrak{f}_\tau$ by the definition of $\mathfrak{f}_\sigma$ and $\mathfrak{f}_\tau$.
$\mathfrak{f}_\sigma$ and $\mathfrak{f}_\tau$ agree on $\mathsf{dom}\, \mathfrak{f}_\sigma$, so $\ulcorner \mathfrak{f}_\tau \urcorner \vDash \ulcorner \mathfrak{f}_\sigma \urcorner$. From $\mathsf{dom}\, \mathfrak{f}_\sigma = \Gamma \setminus \{\alpha_\sigma\}^\downarrow$ and $\mathsf{dom}\, \mathfrak{f}_\tau = \Gamma \setminus \{\alpha_\tau\}^\downarrow$ follows $\{\alpha_\sigma\}^\downarrow \supseteq \{\alpha_\tau\}^\downarrow$, so $\neg\alpha_\sigma \vDash \neg\alpha_\tau$. Also, from $\mathsf{dom}\, \mathfrak{f}_\tau \setminus \mathsf{dom}\, \mathfrak{f}_\sigma \subseteq \{\alpha_\sigma\}^\downarrow$ follows $\ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma \vDash \ulcorner \mathfrak{f}_\tau \urcorner$. As result, $\ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma \vDash \ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau$.
Since σ and τ are distinct, $\mathsf{dom}\, \mathfrak{f}_\tau \not\subseteq \mathsf{dom}\, \mathfrak{f}_\sigma$. Then $\mathsf{dom}\, \mathfrak{f}_\tau \cap \{\alpha_\sigma\}^\downarrow$ is non-empty, and by upward closure $\alpha_\sigma \in \mathsf{dom}\, \mathfrak{f}_\tau$. Since $\lambda(\alpha_\sigma) \cap \Delta_\sigma = \emptyset$ by definition of α_σ, but $\mathsf{ran}\, \mathfrak{f}_\sigma \subseteq \Delta_\sigma$, it follows that $\mathfrak{f}_\tau(\alpha_\sigma) \notin \mathsf{ran}\, \mathfrak{f}_\sigma$. As $\mathsf{ran}\, \mathfrak{f}_\sigma$ is downward closed, $\mathfrak{f}_\tau(\alpha_\sigma) \nvDash \beta$ for all $\beta \in \mathsf{ran}\, \mathfrak{f}_\sigma$. By the Hanoi property and multiplicativity, then

$$\delta := \alpha_\sigma \wedge \mathfrak{f}_\tau(\alpha_\sigma) \wedge \bigwedge_{\beta \in \mathsf{ran}\, \mathfrak{f}_\sigma} \neg\beta$$

is satisfiable, implying $\ulcorner \mathfrak{f}_\sigma \urcorner \nvDash \ulcorner \mathfrak{f}_\tau \urcorner$.
To also prove $\ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau \nvDash \ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma$, we simply show that already $\neg\Delta \cup \{\neg\alpha_\tau, \alpha_\sigma\}$ is satisfiable. Since $\alpha_\tau \vDash \alpha_\sigma$, and Γ is partially ordered, $\alpha_\sigma \nvDash \alpha_\tau$. Then $\alpha_\sigma \wedge \neg\alpha_\tau$, and by the Hanoi property also $\neg\Delta \cup \{\neg\alpha_\tau, \alpha_\sigma\}$, is satisfiable.

(ii) σ is not a prefix of τ and vice versa.
We show $\ulcorner \mathfrak{f}_\sigma \urcorner \wedge \neg\alpha_\sigma \nvDash \ulcorner \mathfrak{f}_\tau \urcorner$; by symmetry reasons, then also $\ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\tau \nvDash \ulcorner \mathfrak{f}_\sigma \urcorner$.
For the proof, let $\sigma = (\beta_1, \ldots, \beta_k)$ and $\tau = (\beta'_1, \ldots, \beta'_m)$. There exists a minimal i, $1 \leq i \leq \min\{k, m\}$ such that $\beta_i \neq \beta'_i$. β_i and β'_i are both minimal in $\lambda(\alpha_{\sigma_{i-1}}) = \lambda(\alpha_{\tau_{i-1}})$, and consequently incomparable. Furthermore, $\beta_i \notin \lambda(\alpha_{\sigma_i})$ by definition of α_{σ_i}. Since now $\lambda(\alpha_{\tau_{i-1}}) \setminus \lambda(\alpha_{\sigma_i})$ contains an element β_i incomparable to β'_i, we can apply **4.(b)** and infer $\alpha_{\tau_{i-1}} \wedge \beta'_i \nvDash \alpha_{\sigma_i}$. Consider now the formula

$$\delta := (\alpha_{\tau_{i-1}} \wedge \beta'_i) \wedge \neg\alpha_{\sigma_i} \wedge \bigwedge_{\substack{\alpha' \in \mathsf{dom}\, \mathfrak{f}_\sigma \\ \alpha' \vDash \alpha_{\sigma_i}}} \neg\alpha' \wedge \bigwedge_{\substack{\alpha' \in \mathsf{dom}\, \mathfrak{f}_\sigma \\ \alpha' \nvDash \alpha_{\sigma_i}}} \neg\mathfrak{f}_\sigma(\alpha').$$

We argue that δ is satisfiable. As $\mathfrak{f}_\tau(\alpha_{\tau_{i-1}}) = \beta'_i$, the conjunction $(\alpha_{\tau_{i-1}} \wedge \beta'_i)$ entails $\neg\ulcorner \mathfrak{f}_\tau \urcorner$. Moreover, $\neg\alpha_{\sigma_i} \vDash \neg\alpha_\sigma$. Then δ itself implies $\neg\ulcorner \mathfrak{f}_\tau \urcorner \wedge \neg\alpha_\sigma \wedge \ulcorner \mathfrak{f}_\sigma \urcorner$, proving the claim.
The first large conjunction is consistent with $(\alpha_{\tau_{i-1}} \wedge \beta'_i)$ since $\alpha_{\tau_{i-1}} \wedge \beta'_i \nvDash \alpha'$ follows from $\alpha_{\tau_{i-1}} \wedge \beta'_i \nvDash \alpha_{\sigma_i}$ and $\alpha' \vDash \alpha_{\sigma_i}$.
For the second large conjunction, consider $\alpha' \in \mathsf{dom}\, \mathfrak{f}_\sigma$ such that $\alpha' \nvDash \alpha_{\sigma_i}$. $\mathfrak{f}_\sigma(\alpha')$ is then defined as some β_j in σ. By definition of $\mathfrak{f}_\sigma$, then $\alpha' \in \{\alpha_{\sigma_{j-1}}\}^\downarrow \setminus \{\alpha_{\sigma_j}\}^\downarrow$. But since $\alpha' \notin \{\alpha_{\sigma_i}\}^\downarrow$, and consequently $\alpha' \notin \{\alpha_{\sigma_i}\}^\downarrow \cup \{\alpha_{\sigma_{i+1}}\}^\downarrow \cup \cdots \cup \{\alpha_\sigma\}^\downarrow$, it holds $j - 1 < i$.
If $j = i$, then $\beta_j = \beta_i$, so β_j is incomparable to β'_i. Then $\beta'_i \wedge \neg\beta_j$, and due to the Hanoi property, $(\alpha_{\tau_{i-1}} \wedge \beta'_i) \wedge \neg\beta_j$ is satisfiable.
If $j < i$, then $\beta_j = \beta'_j$, so $\beta'_i \nvDash \beta'_j$, as τ is a linear extension. Again by the Hanoi property, $(\alpha_{\tau_{i-1}} \wedge \beta'_i) \wedge \neg\beta_j$ is then satisfiable. ◀

The Dynamic Geometry of Interaction Machine: A Call-by-Need Graph Rewriter

Koko Muroya[1] and Dan R. Ghica[2]

1 **University of Birmingham, Birmingham, UK**
k.muroya@cs.bham.ac.uk
2 **University of Birmingham, Birmingham, UK**
d.r.ghica@cs.bham.ac.uk

Abstract

Girard's Geometry of Interaction (GoI), a semantics designed for linear logic proofs, has been also successfully applied to programming languages. One way is to use abstract machines that pass a token in a fixed graph, along a path indicated by the GoI. These token-passing abstract machines are space efficient, because they handle duplicated computation by repeating the same moves of a token on the fixed graph. Although they can be adapted to obtain sound models with regard to the equational theories of various evaluation strategies for the lambda calculus, it can be at the expense of significant time costs. In this paper we show a token-passing abstract machine that can implement evaluation strategies for the lambda calculus, with certified time efficiency. Our abstract machine, called the *Dynamic GoI Machine* (DGoIM), rewrites the graph to avoid replicating computation, using the token to find the redexes. The flexibility of interleaving token transitions and graph rewriting allows the DGoIM to balance the trade-off of space and time costs. This paper shows that the DGoIM can implement call-by-need evaluation for the lambda calculus by using a strategy of interleaving token passing with as much graph rewriting as possible. Our quantitative analysis confirms that the DGoIM with this strategy of interleaving the two kinds of possible operations on graphs can be classified as "efficient" following Accattoli's taxonomy of abstract machines.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases Geometry of Interaction, cost analysis, call-by-need reduction

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.32

1 Introduction

1.1 Token-passing Abstract Machines for λ-calculus

Girard's Geometry of Interaction (GoI) [16] is a semantic framework for linear logic proofs [15]. One way of applying it to programming language semantics is via "token-passing" abstract machines. A term in the λ-calculus is evaluated by representing it as a graph, then passing a token along a path indicated by the GoI. Token-passing GoI decomposes higher-order computation into local token actions, or low-level interactions of simple components. It can give strikingly innovative implementation techniques for functional programs, such as Mackie's *Geometry of Implementation* compiler [18], Ghica's *Geometry of Synthesis* (GoS) high-level synthesis tool [12], and Schöpp's resource-aware program transformation to a low-level language [24]. The interaction-based approach is also convenient for the complexity analysis of programs, e.g. Dal Lago and Schöpp's IntML type system of logarithmic-space evaluation [7], and Dal Lago et al.'s linear dependent type system of polynomial-time evaluation [5, 6].

Fixed-space execution is essential for GoS, since in the case of digital circuits the memory footprint of the program must be known at compile-time, and fixed. Using a restricted version of the call-by-name language Idealised Algol [13] not only the graph, but also the token itself can be given a fixed size. Surprisingly, this technique also allows the compilation of recursive programs [14]. The GoS compiler shows both the usefulness of the GoI as a guideline for unconventional compilation and the natural affinity between its space-efficient abstract machine and call-by-name evaluation. The practical considerations match the prior theoretical understanding of this connection [9].

In contrast, re-evaluating a term by repeating its token actions poses a challenge for call-by-value evaluation because duplicated computation must not lead to repeated evaluation [11, 23, 17, 3]. Moreover, in call-by-value repeating token actions raises the additional technical challenge of avoiding repeating any associated computational effects [22, 21, 4]. A partial solution to this conundrum is to focus on the soundness of the equational theory, while deliberately ignoring the time costs [21]. However, Fernández and Mackie suggest that in a call-by-value scenario, the time efficiency of a token-passing abstract machine could also be improved, by allowing a token to jump along a path, even though a time cost analysis is not given [11].

For us, solving the the problem of creating a GoI-style abstract machine which computes efficiently with evaluation strategies other than call-by-name is a first step in a longer-range research programme. The compilation techniques derived from the GoI can be extremely useful in the case of unconventional computational platforms. But if GoI-style techniques are to be used in a practical setting they need to extend beyond call-by-name, not just correctly but also efficiently.

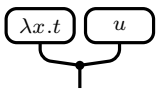
1.2 Interleaving Token Passing with Graph Rewriting

A token jumping, rather than following a path, can be seen as a simple form of short-circuiting that path, which is a simple form of graph-rewriting. This idea first occurs in Mackie's work as a compiler optimisation technique [18] and is analysed in more depth theoretically by Danos and Regnier in the so-called *Interaction Abstract Machine* [9]. More general graph-rewriting-based semantics have been used in a system called *virtual reduction* [8], where rewriting occurs along paths indicated by GoI, but without any token-actions. The most operational presentation of the combination of token-passing and jumping was given by Fernández and Mackie [11]. The interleaving of token actions and rewriting is also found in Sinot's interaction nets [25, 26]. We can reasonably think of the DGoIM as their abstract-machine realisation.

We build on these prior insights by adding more general, yet still efficient, graph-rewriting facilities to the setting of a GoI token-passing abstract machine. We call an abstract machine that interleaves token passing with graph rewriting the *Dynamic GoI Machine* (DGoIM), and we define it as a state transition system with transitions for token passing as well as transitions for graph rewriting. What connects these two kinds of transitions is the token trajectory through the graph, its path. By examining it, the DGoIM can detect redexes and trigger rewriting actions.

Through graph rewriting, the DGoIM reduces sub-graphs visited by the token, avoiding repeated token actions and improving time efficiency. On the other hand, graph rewriting can expand a graph by e.g. copying sub-graphs, so space costs can grow. To control this trade-off of space and time cost, the DGoIM has the flexibility of interleaving token passing with graph rewriting. Once the DGoIM detects that it has traversed a redex, it may rewrite it, but it may also just propagate the token without rewriting the redex.

As a first step in our exploration of the flexibility of this machine, we consider the two extremal cases of interleaving. The first extremal case is "passes-only," in which the DGoIM never triggers graph rewriting, yielding an ordinary token-passing abstract machine. As a typical example, the λ-term $(\lambda x.t)\, u$ is evaluated like this:



1. A token enters the graph on the left at the bottom open edge.
2. A token visits and goes through the left sub-graph $\lambda x.t$.
3. Whenever a token detects an occurrence of the variable x in t, it traverses the right sub-graph u, then returns carrying the resulting value.
4. A token finally exits the graph at the bottom open edge.

Step 3 is repeated whenever term u needs to be re-evaluated. This strategy of interleaving corresponds to call-by-name reduction.

The other extreme is "rewrites-first," in which the DGoIM interleaves token passing with as much, and as early, graph rewriting as possible, guided by the token. This corresponds to both call-by-value and call-by-need reductions, the difference between the two being the trajectory of the token. In the case of call-by-value, the token will enter the graph from the bottom, traverse the left-hand-side sub-graph, which happens to be already a value, then visit sub-graph u even before x is used in a call. While traversing u, it will cause rewrites such that when the token exits, it leaves behind the graph of a machine corresponding to a value v such that u reduces to v. The difference with call-by-need is that the token will visit u only when x is encountered in $\lambda x.t$. In both cases, if repeated evaluation is required then the sub-graph corresponding now to v is copied, so that one copy can be further rewritten, if needed, while the original is kept for later reference.

1.3 Contributions

This work presents a DGoIM model for call-by-need, which can be seen as a case study of the flexibility achieved through controlled interleaving of rewriting and token-passing. This is achieved through a rewriting strategy which turns out to be as natural as the passes-only strategy is for implementing call-by-name. The DGoIM avoids re-evaluation of a sub-term by rewriting any sub-graph visited by a token so that the updated sub-graph represents the evaluation result, but, unlike call-by-value, it starts by evaluating the sub-graph corresponding to the function $\lambda x.t$ first. We chose call-by-need mainly because of the technical challenges it poses. Adapting the technique to call-by-value is a straightforward exercise, and we discuss other alternative in the Conclusion.

We analyse the time cost of the DGoIM with the rewrites-first interleaving, using Accattoli et al.'s general methodology for quantitative analysis [2, 1]. Their method cannot be used "off the shelf," because the DGoIM does not satisfy one of the assumptions used in [1, Section 3]. Our machine uses a more refined transition system, in which several steps correspond to a single one in *loc. cit.*. We overcome this technical difficulty by building a weak simulation of Danvy and Zerny's storeless abstract machine [10] to which the recipe does apply. The result of the quantitative analysis confirms that the DGoIM with the rewrites-first interleaving can be classified as "efficient," following Accattoli's taxonomy of abstract machines introduced in [1].

As we intend to use the DGoIM as a starting point for semantics-directed compilation, this result is an important confirmation that no hidden inefficiencies lurk within the fabric of the rather complex machinery of the DGoIM.

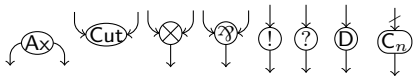


Figure 1 Generators of Graphs.

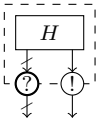


Figure 2 !-box H.

Note. A longer version of this article including all proofs is available as a technical report [20].

2 The Dynamic GoI Machine

The graphs used to construct the DGoIM are essentially MELL proof structures of the multiplicative and exponential fragment of linear logic [15]. They are directed, and built over the fixed set of nodes called "generators" shown in Figure 1.

A C_n-node is annotated by a natural number n that indicates its in-degree, i.e. the number of incoming edges. It generalises a contraction node, whose in-degree is 2, and a weakening node, whose in-degree is 0, of MELL proof structures. In Figure 1, a bunch of n edges is depicted by a single arrow with a strike-out. Graphs must satisfy the well-formedness condition below. Note that, unlike the usual approach [15], we need not assign MELL formulas to edges, nor require a graph to be a valid proof net.

▶ **Definition 1** (well-boxed)**.** A directed graph G built over the generators in Figure 1 is *well-boxed* if:

- it has no incoming edges
- each !-node v in G comes with a sub-graph H of G and an arbitrary number of ?-nodes $\vec{u}$ such that:
  - the sub-graph H (called "!-box") is well-boxed inductively and has at least one outgoing edges
  - the !-node v (called "principal door of H") is the target of one outgoing edge of H
  - the ?-nodes $\vec{u}$ (called "auxiliary doors of H") are the targets of all the other outgoing edges of H
- each ?-node is an auxiliary door of exactly one !-box
- any two distinct !-boxes with distinct principal doors are either disjoint or nested

Note that a !-box might have no auxiliary doors. We use a dashed box to indicate a !-box together with its principal door and its auxiliary doors, as in Figure 2. The auxiliary doors are depicted by a single ?-node with a thick frame and with single incoming and outgoing arrows with strike-outs. Directions of edges are omitted in the rest of the paper, if not ambiguous, to reduce visual clutter.

The DGoIM is formalised as a labelled transition system with two kinds of transitions, namely *pass* transitions $\dashrightarrow$ and *rewrite* transitions $\rightsquigarrow$. Labels of transitions are $\mathsf{b}, \mathsf{s}, \mathsf{o}$ that stand for "beta," "substitution," and "overheads" respectively.

▶ **Definition 2.** Let $\mathcal{L}$ be a fixed countable (infinite) set of *names*. The state of the transition system $s = (\mathbb{G}, p, h, m)$ consists of the following elements:

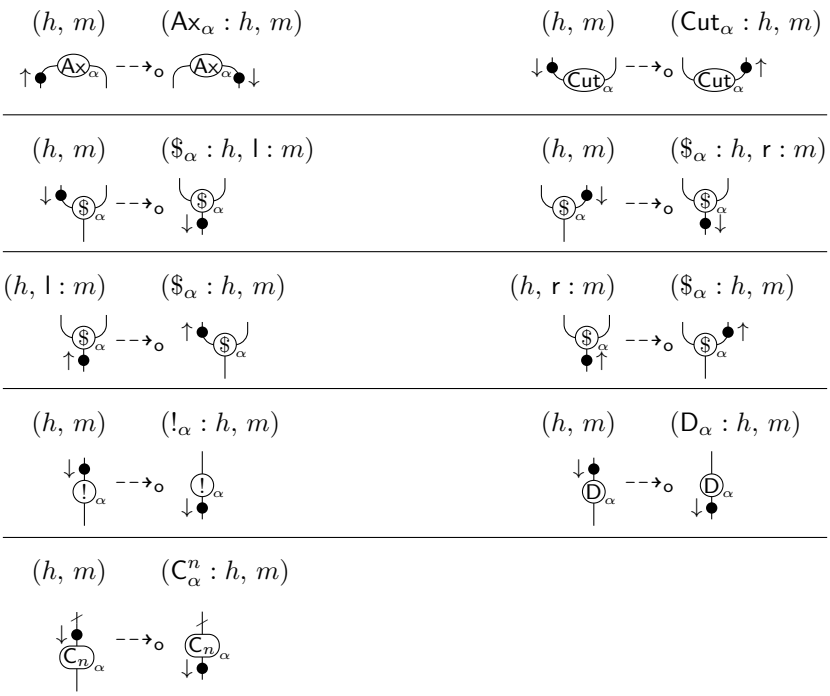


Figure 3 Pass Transitions ($\$ \in \{\otimes, \parr\}$, $n > 0$).

- a *named well-boxed graph* $\mathbb{G} = (G, \ell_G)$, that is a well-boxed graph G with a *naming* l_G that assigns a unique name $\alpha \in \mathcal{L}$ to each node of G
- a pair $p = (e, d)$ called *position*, of an edge e of G and a *direction* $d \in \{\uparrow, \downarrow\}$
- a *history stack* h defined by the grammar below, $\alpha \in \mathcal{L}, n \in \mathbb{N}$:

$$h ::= \Box \mid \mathsf{Ax}_\alpha : h \mid \mathsf{Cut}_\alpha : h \mid \otimes_\alpha : h \mid \parr_\alpha : h \mid !_\alpha : h \mid \mathsf{D}_\alpha : h \mid \mathsf{C}^n_\alpha : h.$$

- a *multiplicative stack* m defined by the BNF grammar $m ::= \Box \mid \mathsf{l} : m \mid \mathsf{r} : m$.

A pass transition $(\mathbb{G}, p, h, m) \dashrightarrow_{\mathsf{o}} (\mathbb{G}, p', h', m')$ changes a position using a multiplicative stack, pushes to a history stack, and keeps a named graph unchanged. All pass transitions have the label o.

Figure 3 shows pass transitions graphically, omitting irrelevant parts of graphs. A position $p = (e, d)$ is represented by a bullet $\bullet$ (called "token") on the edge e together with the direction d. Recall that an edge with a strike-out represents a bunch of edges. The transition in the last line of Figure 3 (where we assume $n > 0$) moves a token from one of the incoming edges of a C_n-node to the outgoing edge of the node. Node names $\alpha \in \mathcal{L}$ are indicated wherever needed.

A rewrite transition $(\mathbb{G}, (e, d), h, m) \rightsquigarrow_{\mathsf{x}} (\mathbb{G}', (e', d), h', m)$ consumes some elements of a history stack, rewrites a sub-graph of a named graph, and updates a position (or, more precisely, its edge). The label x of a rewrite transition $\rightsquigarrow_{\mathsf{x}}$ is either b, s or o. Figure 4 shows rewrite transition in the same manner as Figure 3. Multiplicative stacks are not present in the figure since they are irrelevant. The $\sharp$-node represents some arbitrary node (incoming edges omitted). We can see that no rewrite transition breaks the well-boxed-ness of a graph.

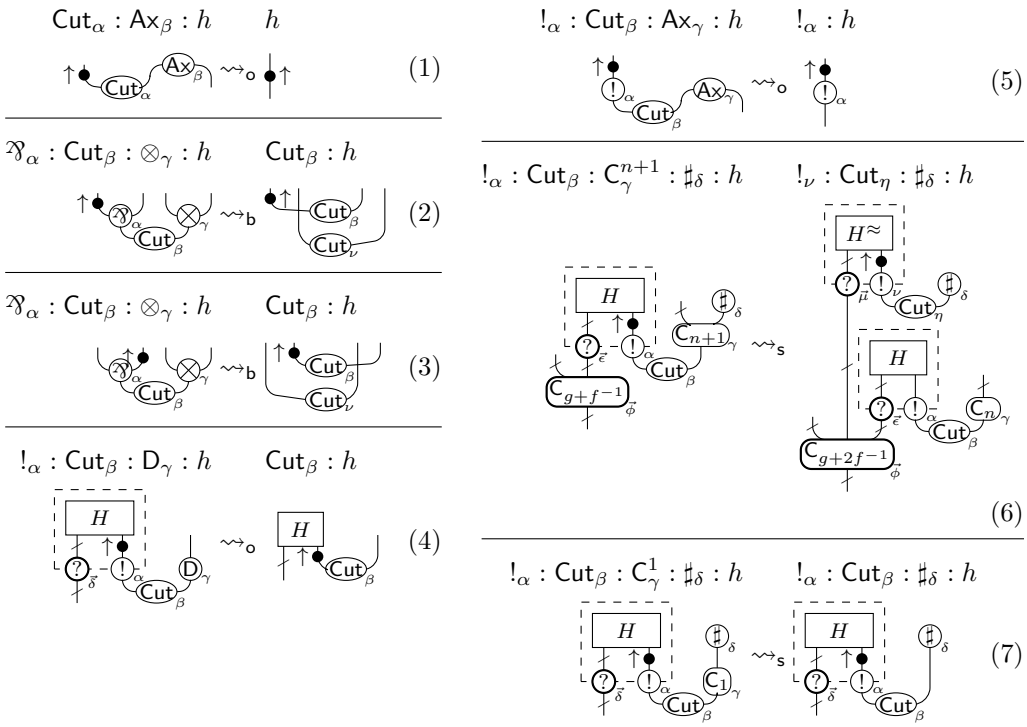


Figure 4 Rewrite Transitions ($n > 0$).

The rewrite transitions (1), (2), (3), and (4) are exactly taken from MELL cut elimination [15]. The rewrite transition (5) is a variant of (1). It acts on a connected pair of a Cut-node and an Ax-node that arises as a result of the transition (6) or (7) but cannot be rewritten by the transition (1). These transitions (6) and (7) are inspired by the MELL cut elimination process for (binary) contraction nodes; note that we assume $n > 0$ in Figure 4.

The rewrite transition (6) in Figure 4 deserves further explanation. The sub-graph $H^{\approx}$ is a copy of the !-box H where all the names are replaced with fresh ones. The thick $\mathsf{C}_{g+f^{-1}}$-node and $\mathsf{C}_{g+2f^{-1}}$-node represent families $\{\mathsf{C}_{g(j)+f^{-1}(j)}\}_{j=0}^{m}, \{\mathsf{C}_{g(j)+2f^{-1}(j)}\}_{j=0}^{m}$, of C-nodes respectively. They are connected to ?-nodes $\vec{\epsilon} = \epsilon_0, \ldots, \epsilon_l$ and $\vec{\mu} = \mu_0, \ldots, \mu_l$ in such a way that:

- the natural numbers l, m satisfy $l \geq m$, and come with a surjection $f\colon \{0, \ldots, l\} \twoheadrightarrow \{0, \ldots, m\}$ and a function $g\colon \{0, \ldots, m\} \to \mathbb{N}$ to the set $\mathbb{N}$ of natural numbers
- each ?-node ϵ_i and each ?-node μ_i are both connected to the C-node $\phi_{f(i)}$
- each C-node ϕ_j has $g(j)$ incoming edges whose source is none of the ?-nodes $\vec{\epsilon}, \vec{\mu}$.

Some rewrite transitions introduce new nodes to a graph. We *require* that the uniqueness of names throughout a whole graph is not violated by these transitions. Under this requirement, the introduced names $\nu, \vec{\mu}$ and the renaming $H^{\approx}$ in Figure 4 can be arbitrary.

▶ **Definition 3.** We call a state $((G, \ell_G), p, h, m)$ *rooted at* e_0 for an open (outgoing) edge e_0 of G, if there exists a finite sequence $((G, \ell_G), (e_0, \uparrow), \square, \square) \dashrightarrow^* ((G, \ell_G), p, h, m)$ of pass transitions such that the position p appears only last in the sequence.

Lemma 4(1) below implies that, the DGoIM can determine whether a rewrite transition is possible at a rooted state by only examining a history stack. The rooted property is preserved by transitions.

▶ **Lemma 4** (rooted states). *Let* $((G, \ell_G), (e, d), h, m)$ *be a rooted state at* e_0 *with a (finite) sequence* $((G, \ell_G), (e_0, \uparrow), \Box, \Box) \dashrightarrow^* ((G, \ell_G), (e, d), h, m)$.

1. *The history stack represents an (undirected and possibly cyclic) path of graph* G *connecting edges* e_0 *and* e.
2. *If a transition* $((G, \ell_G), (e, d), h, m)$ $(\dashrightarrow \cup \rightsquigarrow)$ $((G', \ell_{G'}), p', h', m')$ *is possible, the open edges of* G' *are bijective to those of* G*, and the state* $((G', \ell_{G'}), p', h', m')$ *is rooted at the open edge corresponding to* e_0.

2.1 Cost Analysis of the DGoIM

The time cost of updating stacks is constant, as each transition changes only a fixed number of top elements of stacks. Updating a position is local and needs constant time, as it does not require searching beyond the next edge in the graph from the current edge. We can conclude all pass transitions take constant time.

We estimate the time cost of rewrite transitions by counting updated nodes. The rewrite transitions (1)–(3) involve a fixed number of nodes, and transition (7) eliminates one C_1-node. Only transitions (4) and (6) have non-constant time cost. The number of doors deleted in transition (4) can be arbitrary, and so is the number of nodes introduced in transition (6).

Pass transitions and rewrite transitions are separately deterministic (up to the choice of new names). However, both a pass transition and a rewrite transition are possible at some states. We here opt for the following "rewrites-first" way to interleave pass transitions with as much rewrite transitions as possible:

$$s \twoheadrightarrow_{\mathsf{x}} s' \overset{\text{Definition}}{\Longleftrightarrow} \begin{cases} s \rightsquigarrow_{\mathsf{x}} s' & (\text{if } \rightsquigarrow_{\mathsf{x}} \text{ possible}) \\ s \dashrightarrow_{\mathsf{x}} s' & (\text{if only } \dashrightarrow_{\mathsf{x}} \text{ possible}). \end{cases}$$

The DGoIM with this strategy yields a deterministic labelled transition system $\twoheadrightarrow$ up to the choice of new names in rewrite transitions. We denote it by $\text{DGoIM}_{\twoheadrightarrow}$, making the strategy explicit. Note that there can be other strategies of interleaving although we do not explore them here.

Space. Before we conclude, several considerations about space cost analysis. Space costs are generally bound by time costs, so from our analysis there is an implicit guarantee that space usage will not explode. But if a more refined space cost analysis is desired, the following might prove to be useful.

The space required in implementing a named well-boxed graph is bounded by the number of its nodes. The number of edges is linear in the number of nodes, because each generator has a fixed out-degree and every edge of a well-boxed graph has its source.

Additionally a !-box can be represented by associating its auxiliary doors to its principal door. This adds connections between doors to a graph that are as many as ?-nodes. It enables the DGoIM to identify nodes of a !-box by following edges from its principal and auxiliary doors. Nodes in a !-box that are not connected to doors can be ignored, since these nodes are never visited by a token (i.e. pointed by a position) as long as the DGoIM acts on rooted states.

Only the rewrite transition (6) can increase the number of nodes of a graph by copying a !-box with its doors. Rewrite transitions can copy !-boxes and eliminate the !-box structure, but they never create new !-boxes or change existing ones. This means that, in a sequence of transitions that starts with a graph G, any !-boxes copied by the rewrite transition (6) are sub-graphs of the graph G. Therefore the number of nodes of a graph increases linearly in the number of transitions.

Terms	$t ::= x \mid \lambda x.t \mid t\,t \mid t[x \leftarrow t]$	Pure terms	$\bar{t} ::= x \mid \lambda x.\bar{t} \mid \bar{t}\,\bar{t}$
Values	$v ::= \lambda x.t$	Pure values	$\overline{v} ::= \lambda x.\bar{t}$
Evaluation contexts	$E ::= \langle\cdot\rangle \mid E\,\bar{t} \mid E[x \leftarrow \bar{t}] \mid E\langle x\rangle[x \leftarrow E]$		
Substitution contexts	$A ::= \langle\cdot\rangle \mid A[x \leftarrow \bar{t}]$		

$$(\bar{t}\,\overline{u},\, E)_{term} \to_{\mathsf{o}} (\bar{t},\, E\langle\langle\cdot\rangle\,\overline{u}\rangle)_{term} \tag{8}$$

$$(x,\, E_1\langle E_2[x \leftarrow \bar{t}]\rangle)_{term} \to_{\mathsf{o}} (\bar{t},\, E_1\langle E_2\langle x\rangle[x \leftarrow \langle\cdot\rangle]\rangle)_{term} \tag{9}$$

$$(\overline{v},\, E)_{term} \to_{\mathsf{o}} (\overline{v},\, E)_{ctxt} \tag{10}$$

$$(\lambda x.\bar{t},\, E\langle A\,\overline{u}\rangle)_{ctxt} \to_{\mathsf{b}} (\bar{t},\, E\langle A\langle\langle\cdot\rangle[x \leftarrow \overline{u}]\rangle\rangle)_{term} \tag{11}$$

$$(\overline{v},\, E_1\langle E_2\langle x\rangle[x \leftarrow A]\rangle)_{ctxt} \to_{\mathsf{s}} (\overline{v}^{\approx},\, E_1\langle A\langle E_2[x \leftarrow \overline{v}]\rangle\rangle)_{ctxt} \quad (\text{if } x \in \mathrm{FV}_{\emptyset}(E_2)) \tag{12}$$

$$(\overline{v},\, E_1\langle E_2\langle x\rangle[x \leftarrow A]\rangle)_{ctxt} \to_{\mathsf{s}} (\overline{v},\, E_1\langle A\langle E_2\rangle\rangle)_{ctxt} \quad (\text{if } x \notin \mathrm{FV}_{\emptyset}(E_2)) \tag{13}$$

Figure 5 Call-by-need Storeless Abstract Machine (SAM).

Elements of history stacks and multiplicative stacks, as well as a position, are essentially pointers to nodes. Because each pass/rewrite transition adds at most one element to each stack, the lengths of stacks also grow linearly in the number of transitions.

3 Weak Simulation of the Call-by-Need Storeless Abstract Machine

We show the DGoIM$_{\to}$ implements call-by-need evaluation by building a weak simulation of the call-by-need Storeless Abstract Machine (SAM) defined in Figure 5. It simplifies Danvy and Zerny's storeless machine [10, Figure 8] and accommodates a partial mechanism of garbage collection (namely, transition (13)). We will return to a discussion of garbage collection at the end of this section.

The SAM is a labelled transition system between *configurations* $(\bar{t}, E)$. They are classified into two groups, namely *term* configurations and *context* configurations, that are indicated by annotations *term*, *ctxt* respectively. Pure terms (resp. pure values) are terms (resp. values) that contain no explicit substitutions $t[x \leftarrow u]$; we sometimes omit the word "pure" and the overline in denotation, if unambiguous.

Each evaluation context E contains exactly one open hole $\langle\cdot\rangle$, and replacing it with a term t (or an evaluation context E') yields a term $E\langle t\rangle$ (or an evaluation context $E\langle E'\rangle$) called *plugging*. In particular an evaluation context $E'\langle x\rangle[x \leftarrow E]$ replaces the open hole of E' with x and keeps the open hole of E.

Labels of transitions are the same as those used for the DGoIM (i.e. b, s and o). The transition (11), with the label b, corresponds to the β-reduction where evaluation and substitution of function arguments are delayed. Substitution happens in the transitions (12) and (13), with the label s, that replaces exactly one occurrence of a variable. The other transitions with the label o, namely $(\bar{t}, E) \to_{\mathsf{o}} (\bar{t'}, E')$, search a redex by rearranging a configuration. The two pluggings $E\langle\bar{t}\rangle$ and $E'\langle\bar{t'}\rangle$ indeed yield exactly the same term.

We characterise "free" variables using multisets of variables. Multisets make explicit how many times a variable is duplicated in a term (or an evaluation context). This duplication of information is later used in translating terms to graphs.

▶ **Notation** (multiset). *A multiset* $\mathbf{x} := [x, \ldots, x]$ *consists of a finite number of* x*s. The multiplicity of* x *in a multiset* M *is denoted by* $M(x)$*. We write* $x \in^k M$ *if* $M(x) = k$*,* $x \in M$ *if* $M(x) > 0$ *and* $x \notin M$ *if* $M(x) = 0$*. A multiset* M *comes with its support set* $\mathrm{supp}(M)$*.*

For two multisets M and M', their sum and difference are denoted by $M + M'$ and $M - M'$ respectively. Removing all *x from a multiset M yields the multiset $M\backslash x$, e.g. $[x, x, y]\backslash x = [y]$.*

Each term t and each evaluation context E are respectively assigned multisets of variables $\mathrm{FV}(t), \mathrm{FV}_M(E)$, with M a multiset of variables. The multisets FV are defined inductively as follows.

$$\begin{aligned}
\mathrm{FV}(x) &:= [x],\\
\mathrm{FV}(\lambda x.t) &:= \mathrm{FV}(t)\backslash x,\\
\mathrm{FV}(t\,u) &:= \mathrm{FV}(t) + \mathrm{FV}(u),\\
\mathrm{FV}(t[x \leftarrow u]) &:= (\mathrm{FV}(t)\backslash x) + \mathrm{FV}(u).\\
\mathrm{FV}_M(\langle\cdot\rangle) &:= M,\\
\mathrm{FV}_M(E\,t) &:= \mathrm{FV}_M(E) + \mathrm{FV}(t),\\
\mathrm{FV}_M(E[x \leftarrow t]) &:= (\mathrm{FV}_M(E))\backslash x + \mathrm{FV}(t),\\
\mathrm{FV}_M(E'\langle x\rangle[x \leftarrow E]) &:= (\mathrm{FV}_{[x]}(E'))\backslash x + \mathrm{FV}_M(E).
\end{aligned}$$

The following equations can be proved by a straightforward induction on E.

▶ **Lemma 5** (decomposition)**.**

$$\begin{aligned}
\mathrm{FV}(E\langle t\rangle) &= \mathrm{FV}_{\mathrm{FV}(t)}(E)\\
\mathrm{FV}_M(E\langle E'\rangle) &= \mathrm{FV}_{\mathrm{FV}_M(E')}(E)
\end{aligned}$$

A variable x is *bound* in a term t if it appears in the form of $\lambda x.u$ or $u[x \to u']$. A variable x is *captured* in an evaluation context E if it appears in the form of $E'[x \leftarrow \overline{t}]$ (but not in the form of $E'\langle x\rangle[x \leftarrow E'']$). Transitions (12) and (13) depend on whether or not the bound variable x appears in the evaluation context E_2. If the variable x appears, the value $\overline{v}$ is kept for later use and its copy $\overline{v}^{\approx}$ is substituted for x. If not, the value $\overline{v}$ itself is substituted for x.

The SAM does not assume α-equivalence, but explicitly deals with it in copying a value. The copy $\overline{v}^{\approx}$ has all its bound variables replaced by distinct fresh variables (i.e. distinct variables that do not appear in a whole configuration). This implies that the SAM is deterministic up to the choice of new variables introduced in copying.

A term t is *closed* if $\mathrm{FV}(t) = \emptyset$; and is *well-named* if each variable gets bound at most once in t, and each bound variable x in t satisfies $x \notin \mathrm{FV}(t)$. An *initial* configuration is a term configuration $(\overline{t_0}, \langle\cdot\rangle)_{term}$ where $\overline{t_0}$ is closed and well-named. A finite sequence of transitions from an initial configuration is called an *execution*. A *reachable* configuration $(\overline{t}, E)$, that is a configuration coming with an execution from some initial configuration to itself, satisfies the following invariant properties.

▶ **Lemma 6** (reachable configurations)**.** *Let $(\overline{t}, E)$ be a reachable configuration from an initial configuration $(\overline{t_0}, \langle\cdot\rangle)_{term}$. The term $\overline{t}$ is a sub-term of the initial term $\overline{t_0}$ up to α-equivalence, and the plugging $E\langle\overline{t}\rangle$ is closed and well-named.*

The proof is by induction on the length of the execution.

We now conclude with a brief consideration on *garbage collection*. Transition (13) eliminates an explicit substitution and therefore implements a partial mechanism of garbage collection. The mechanism is partial because only an explicit substitution that is looked up in an execution can be eliminated, as illustrated below. The explicit substitution $[x \leftarrow \lambda z.z]$ is eliminated in $((\lambda x.x)\,(\lambda z.z), \langle\cdot\rangle)_{term} \to^* (\lambda z.z, \langle\cdot\rangle)_{ctxt}$, but not in $((\lambda x.\lambda y.y)\,(\lambda z.z), \langle\cdot\rangle)_{term} \to^*$

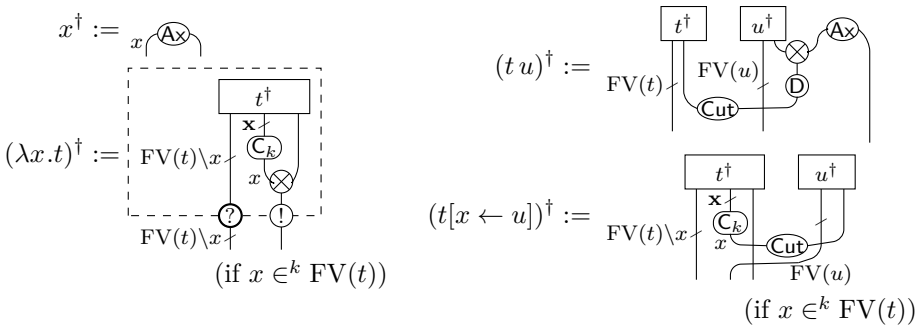


Figure 6 Inductive Translation $(\cdot)^\dagger$ of Terms to Well-boxed Graphs.

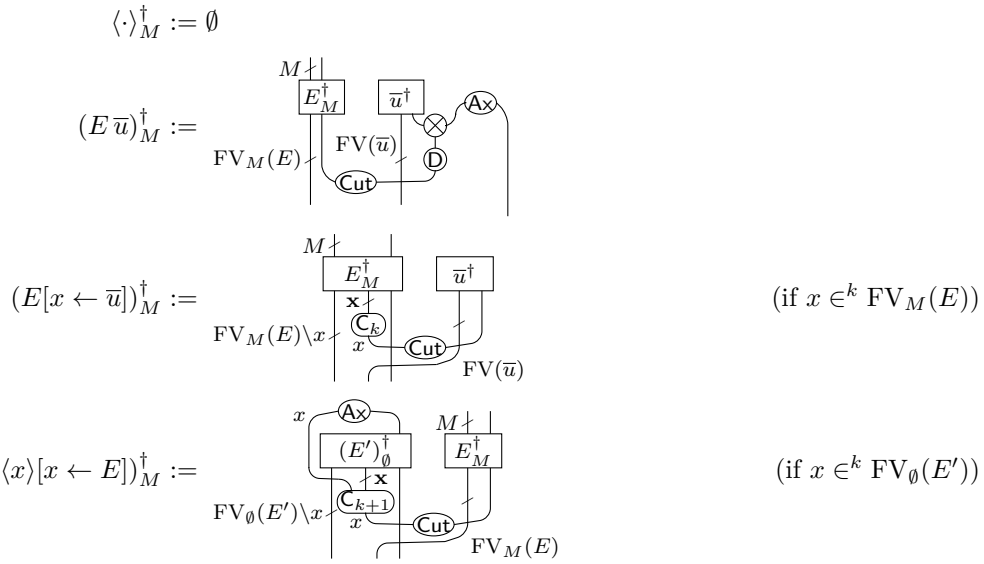
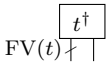
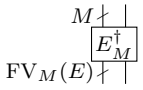


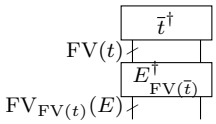
Figure 7 Inductive Translation $(\cdot)^\dagger_M$ of Evaluation Contexts to Graphs.

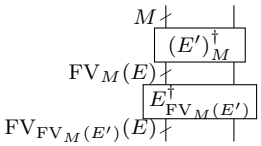
$(\lambda y.y, \langle\cdot\rangle[x \leftarrow \lambda z.z])_{ctxt}$, because the bound variable x does not occur. We incorporate this partial garbage collection to clarify the use of the rewrite transitions (6) and (7).

We can now define a weak simulation using translations of terms and evaluation contexts. The translations $(\cdot)^\dagger$ are inductively defined in Figure 6 and Figure 7. What underlies them is the so-called "call-by-value" translation of intuitionistic logic to linear logic. This translates all and only values to !-boxes that can be copied by rewrite transitions.

The translation  of a term t is a well-boxed graph, where some edges are annotated with variables to help understanding. We continue representing a bunch of edges by a single edge and a strike-out, with annotations denoted by a multiset, and a bunch of nodes by a single thick node. The translation  of an evaluation context E, given a multiset M of variables, is not a well-boxed graph because it has incoming edges. Lemma 7 is analogous to Lemma 5; their proof is by straightforward induction on E.

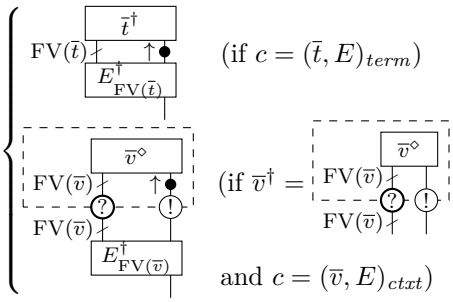
▶ **Lemma 7** (decomposition).

$(E\langle t\rangle)^\dagger =$ 

$(E\langle E'\rangle)^\dagger_M =$ 

The translations $(\cdot)^\dagger$ are lifted to a binary relation between reachable configurations of the SAM and rooted states of the DGoIM$_{\multimap}$.

▶ **Definition 8** (binary relation $\preceq$). A reachable configuration c and a state $((G,\ell_G),p,h,m)$ satisfies $c \preceq ((G,\ell_G),p,h,m)$ if and only if ℓ_G is an arbitrary naming, $((G,\ell_G),p,h,m)$ is rooted at the unique open edge of G, and

$(G,p) =$ 

Note that the graph G in the above definition has exactly one open edge, because it is equal to the translation $E\langle\overline{t}\rangle^\dagger$ (Lemma 7) and the plugging $E\langle\overline{t}\rangle$ is closed (Lemma 6).

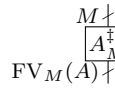
The binary relation $\preceq$ gives a weak simulation, as stated below. It is weak in Milner's sense [19], where transitions with the label o are regarded as internal. We can conclude from Theorem 9 below that the DGoIM$_{\multimap}$ soundly implements the call-by-need evaluation.

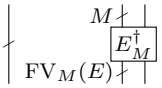
▶ **Theorem 9** (weak simulation). *Let a configuration c and a state s satisfy $c \preceq s$.*

1. *If a transition $c \to_{\mathsf{b}} c'$ of the SAM is possible, there exists a sequence $s \multimap^2_{\mathsf{o}} \multimap_{\mathsf{b}} \multimap_{\mathsf{o}} s'$ such that $c' \preceq s'$.*
2. *If a transition $c \to_{\mathsf{s}} c'$ of the SAM is possible, there exists a sequence $s \multimap_{\mathsf{s}} \multimap_{\mathsf{o}} s'$ such that $c' \preceq s'$.*
3. *If a transition $c \to_{\mathsf{o}} c'$ of the SAM is possible, there exists a sequence $s \multimap^N_{\mathsf{o}} s'$ such that $0 < N \leq 4$ and $c' \preceq s'$.*
4. *No transition $\multimap$ is possible at the state s' if $c' = (\overline{v}, A)_{ctxt}$.*

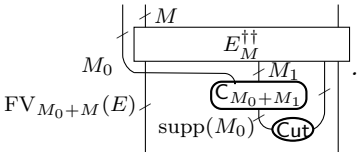
They key ingredients for the proof are the decomposition properties in Lemma 7 as well as the other decomposition properties from the following Lemma 10. Application relies on reachable configurations being closed and well-named, in the sense of Lemma 6.

▶ **Lemma 10** (decomposition). *Let M_0, M be multisets of variables.*

1. *The translation $A^\dagger_M$ of a substitution context A has a unique decomposition* 

.
2. *If no variables in M_0 are captured in an evaluation context E, the translation $E^\dagger_{M_0+M}$ is equal to the graph* 

.

3. *If each variable in M_0 is captured in an evaluation context E exactly once, the translation $E^{\dagger}_{M_0+M}$ has a unique decomposition* 

. The multiset M_1 satisfies $\mathrm{supp}(M_1) \subseteq \mathrm{supp}(M_0)$, and the thick $\mathsf{C}_{M_0+M_1}$-node represents a family $\{\mathsf{C}_{M_0(x)+M_1(x)}\}_{x\in\mathrm{supp}(M_0)}$ of C-nodes.

The proofs for 1. and 2. are by straightforward inductions on A and E respectively. The proof for 3. is by induction on the dimension of M_0, i.e. the size of the support set $\mathrm{supp}(M_0)$.

4 Time Cost Analysis of Rewrites-First Interleaving

Our time cost analysis of the DGoIM$_{\multimap}$ follows Accattoli's recipe, described in [2, 1], of analysing complexity of abstract machines. This section recalls the recipe and explains how it applies to the DGoIM$_{\multimap}$.

The time cost analysis focuses on how efficiently an abstract machine implements an evaluation strategy. In other words, we are not interested in minimising the number of β-reduction steps simulated by an abstract machine. Our interest is in making the number of transitions of an abstract machine "reasonable," compared to the number of necessary β-reduction steps determined by a given evaluation strategy.

Accattoli's recipe assumes that an abstract machine has three groups of transitions: 1) "β-transitions" that correspond to β-reduction in which substitution is delayed, 2) transitions perform substitution, and 3) other "overhead" transitions. We incorporate this classification using the labels $\mathsf{b}, \mathsf{s}, \mathsf{o}$ of transitions.

Another assumption of the recipe is that, each step of β-reduction is simulated by a single transition of an abstract machine, and so is substitution of each occurrence of a variable. This is satisfied by many known abstract machines including the SAM, however not by the DGoIM$_{\multimap}$. The DGoIM$_{\multimap}$ has "finer" transitions and can take several transitions to simulate a single step of reduction (hence a single transition of the SAM, as we can observe in Theorem 9). In spite of this mismatch we can still follow the recipe, thanks to the weak simulation $\preceq$. It discloses what transitions of the DGoIM exactly correspond to β-reduction and substitution, and gives a concrete number of overhead transitions that the DGoIM$_{\multimap}$ needs to simulate β-reduction and substitution. The recipe for the time cost analysis is:

1. Examine the number of transitions, by means of the size of input and the number of β-transitions.
2. Estimate time cost of single transitions.
3. Derive a bound of the overall execution time cost.
4. Classify an abstract machine according to its execution time cost.

Consider now the following taxonomy of abstract machines introduced in [1].

▶ **Definition 11** (classes of abstract machines [1, Definition 7.1])**.** **1.** An abstract machine is *efficient* if its execution time cost is linear in both the input size and the number of β-transitions.

2. An abstract machine is *reasonable* if its execution time cost is polynomial in the input size and the number of β-transitions.
3. An abstract machine is *unreasonable* if it is not reasonable.

The input size in our case is given by the *size* $|t|$ of a term t, inductively defined by:

$$|x| := 1 \qquad |\lambda x.t| := |t| + 1$$
$$|t\,u| := |t| + |u| + 1 \qquad |t[x \leftarrow u]| := |t| + |u| + 1.$$

Given a sequence r of transitions (of either the SAM or the DGoIM$_{\triangleright}$), we denote the number of transitions with a label x in r by $|r|_{\mathsf{x}}$. Since we use the fixed set $\{\mathsf{b}, \mathsf{s}, \mathsf{o}\}$ of labels, the length $|r|$ of the sequence r is equal to the sum $|r|_{\mathsf{b}} + |r|_{\mathsf{s}} + |r|_{\mathsf{o}}$.

We first estimate the number of transitions of the SAM, and then derive estimation for the DGoIM$_{\triangleright}$.

▶ **Lemma 12** (quantitative bounds for SAM). *Each execution e from an initial configuration $(\overline{t_0}, E)_{term}$, comes with inequalities: $|e|_{\mathsf{s}} \leq |e|_{\mathsf{b}}$ and $|e|_{\mathsf{o}} \leq |\overline{t_0}| \cdot (5 \cdot |e|_{\mathsf{b}} + 2) + (3 \cdot |e|_{\mathsf{b}} + 1)$.*

The proof is analogous to the discussion in [2, Section 11].

Combining these bounds for the SAM with the weak simulation $\preceq$, we can estimate the number of transitions of the DGoIM$_{\triangleright}$ as below.

▶ **Proposition 13** (quantitative bounds for DGoIM$_{\triangleright}$). *Let $r\colon s_0 \to_{\triangleright}^* s$ be a sequence of transitions of the DGoIM$_{\triangleright}$. If there exists an execution $(\overline{t_0}, \langle\cdot\rangle)_{term} \to^* (\bar{t}, E)$ of the SAM such that $s_0 \preceq (\overline{t_0}, \langle\cdot\rangle)_{term}$ and $s \preceq (\bar{t}, E)$, the sequence r comes with inequalities $|r|_{\mathsf{s}} \leq |r|_{\mathsf{b}}$ and $|r|_{\mathsf{o}} \leq 4 \cdot |\overline{t_0}| \cdot (5 \cdot |r|_{\mathsf{b}} + 2) + (16 \cdot |r|_{\mathsf{b}} + 4)$.*

This is a direct consequence of Lemma 12 and Theorem 9.

We already discussed time cost of single transitions of the DGoIM in Section 2.1. It is worth noting that the discussion in Section 2.1 is independent of any particular choice of a rewriting and token-passing interleaving strategy.

Theorem 14 below gives a bound of execution time cost of the DGoIM$_{\triangleright}$. We can conclude that, according to Accattoli's taxonomy (see Definition 11), the DGoIM$_{\triangleright}$ is "efficient" as an abstract machine for the call-by-need evaluation.

▶ **Theorem 14** (time cost). *Let C, D be fixed natural numbers, and $r\colon s_0 \to_{\triangleright}^* s$ be a sequence of transitions of the DGoIM$_{\triangleright}$. If there exists an execution $(\overline{t_0}, \langle\cdot\rangle)_{term} \to^* (\bar{t}, E)$ of the SAM such that $s_0 \preceq (\overline{t_0}, \langle\cdot\rangle)_{term}$ and $s \preceq (\bar{t}, E)$, the total time cost $T(r)$ of the sequence r satisfies:*

$$T(r) = \mathcal{O}((|\overline{t_0}| + C) \cdot (|r|_{\mathsf{b}} + D)).$$

▶ **Corollary 15.** *The DGoIM$_{\triangleright}$ is an efficient abstract machine, in the sense of Definition 11.*

5 Conclusions

We introduced the DGoIM, which can interleave token passing with graph rewriting informed by the trajectory of the token. We focused on the rewrites-first interleaving and proved that it enables the DGoIM to implement the call-by-need evaluation strategy. The quantitative analysis of time cost certifies that the DGoIM$_{\triangleright}$ gives an "efficient" implementation in the sense of Accattoli's classification. The proof of Theorem 14 pointed out that eliminating and copying !-boxes are the two main sources of time expenditure. Our results are built on top of a weak simulation of the SAM, that relates several transitions of the DGoIM to each computational task such as β-reduction and substitution.

The main feature of the DGoIM is the flexible combination of interaction and rewriting. We here briefly discuss how the flexibility can enable the DGoIM to implement evaluation strategies other than the call-by-need.

As mentioned in Section 1.2, the passes-only interleaving can yield an ordinary token-passing abstract machine that is known to implement the call-by-name evaluation. We note that the DGoIM presented in Section 2 is only the part relevant to the rewrites-first interleaving. We omitted some pass transitions and data structures carried by a token, that are known in ordinary token-passing abstract machines but irrelevant to the rewrites-first interleaving. For example Figure 3 does not show pass transitions that let a token go through auxiliary doors of a !-box, because in the rewrites-first interleaving, auxiliary doors are eliminated as soon as a token visits their corresponding principal door. Accordingly a token does not carry so-called "exponential signatures" that make sure the token enters and exits !-boxes appropriately.

The only difference between the call-by-need and the call-by-value evaluations lies in when function arguments are evaluated. In the DGoIM, this corresponds to changing a trajectory of a token so that it visits function arguments immediately after it detects function application. Therefore, to implement the call-by-value evaluation, the DGoIM can still use the rewrites-first interleaving, but it should use a modified set of pass transitions. Further refinements, not only of the evaluation strategies but also of the graph representation could yield even more efficient implementation, such as *full lazy evaluation*, as hinted in [25].

Our final remarks concern programming features that have been modelled using token-passing abstract machines. Ground-type constants are handled by attaching memories to either nodes of a graph or a token, in e.g. [18, 17, 3] – this can be seen as a simple form of graph rewriting. Algebraic effects are also accommodated using memories attached to nodes of a graph in token machines [17], but their treatment would be much simplified in the DGoIM as effects are evaluated out of the term via rewriting.

Acknowledgements. We are grateful to Ugo Dal Lago and anonymous reviewers for encouraging and insightful comments on earlier versions of this work.

References

1 Beniamino Accattoli. The complexity of abstract machines. In *WPTE 2016*, volume 235 of *EPTCS*, pages 1–15, 2017.
2 Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In *ICFP 2014*, pages 363–376. ACM, 2014.
3 Ugo Dal Lago, Claudia Faggian, Benoît Valiron, and Akira Yoshimizu. Parallelism and synchronization in an infinitary context. In *LICS 2015*, pages 559–572. IEEE, 2015.
4 Ugo Dal Lago, Claudia Faggian, Benoît Valiron, and Akira Yoshimizu. The Geometry of Parallelism: classical, probabilistic, and quantum effects. In *POPL 2017*, pages 833–845. ACM, 2017.
5 Ugo Dal Lago and Marco Gaboardi. Linear dependent types and relative completeness. In *LICS 2011*, pages 133–142. IEEE Computer Society, 2011.
6 Ugo Dal Lago and Barbara Petit. Linear dependent types in a call-by-value scenario. In *PPDP 2012*, pages 115–126. ACM, 2012.
7 Ugo Dal Lago and Ulrich Schöpp. Computation by interaction for space-bounded functional programming. *Inf. Comput.*, 248:150–194, 2016.
8 Vincent Danos and Laurent Regnier. Local and asynchronous beta-reduction (an analysis of Girard's execution formula). In *LICS 1993*, pages 296–306. IEEE Computer Society, 1993.
9 Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal lambda-machines. *Elect. Notes in Theor. Comp. Sci.*, 3:40–60, 1996.

10 Olivier Danvy and Ian Zerny. A synthetic operational account of call-by-need evaluation. In *PPDP 2013*, pages 97–108. ACM, 2013.

11 Maribel Fernández and Ian Mackie. Call-by-value lambda-graph rewriting without rewriting. In *ICGT 2002*, volume 2505 of *LNCS*, pages 75–89. Springer, 2002.

12 Dan R. Ghica. Geometry of Synthesis: a structured approach to VLSI design. In *POPL 2007*, pages 363–375. ACM, 2007.

13 Dan R. Ghica and Alex Smith. Geometry of Synthesis III: resource management through type inference. In *POPL 2011*, pages 345–356. ACM, 2011.

14 Dan R. Ghica, Alex Smith, and Satnam Singh. Geometry of Synthesis IV: compiling affine recursion into static hardware. In *ICFP*, pages 221–233, 2011.

15 Jean-Yves Girard. Linear logic. *Theor. Comp. Sci.*, 50:1–102, 1987.

16 Jean-Yves Girard. Geometry of Interaction I: interpretation of system F. In *Logic Colloquium 1988*, volume 127 of *Studies in Logic & Found. Math.*, pages 221–260. Elsevier, 1989.

17 Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. Memoryful Geometry of Interaction: from coalgebraic components to algebraic effects. In *CSL-LICS 2014*, pages 52:1–52:10. ACM, 2014.

18 Ian Mackie. The Geometry of Interaction machine. In *POPL 1995*, pages 198–208. ACM, 1995.

19 Robin Milner. *Communication and concurrency*. PHI Series in Computer Science. Prentice Hall, 1989.

20 Koko Muroya and Dan R. Ghica. The dynamic geometry of interaction machine: A call-by-need graph rewriter. *CoRR*, arXiv:1703.10027, 2017. URL: `https://arxiv.org/abs/1703.10027`.

21 Koko Muroya, Naohiko Hoshino, and Ichiro Hasuo. Memoryful Geometry of Interaction II: recursion and adequacy. In *POPL 2016*, pages 748–760. ACM, 2016.

22 Ulrich Schöpp. Computation-by-interaction with effects. In *APLAS 2011*, volume 7078 of *Lect. Notes Comp. Sci.*, pages 305–321. Springer, 2011.

23 Ulrich Schöpp. Call-by-value in a basic logic for interaction. In *APLAS 2014*, volume 8858 of *Lect. Notes Comp. Sci.*, pages 428–448. Springer, 2014.

24 Ulrich Schöpp. Organising low-level programs using higher types. In *PPDP 2014*, pages 199–210. ACM, 2014.

25 François-Régis Sinot. Call-by-name and call-by-value as token-passing interaction nets. In *TLCA 2005*, volume 3461 of *Lect. Notes Comp. Sci.*, pages 386–400. Springer, 2005.

26 François-Régis Sinot. Call-by-need in token-passing nets. *Math. Struct. in Comp. Sci.*, 16(4):639–666, 2006.

On Supergraphs Satisfying CMSO Properties*

Mateus de Oliveira Oliveira

Department of Informatics, University of Bergen, Bergen, Norway
mateus.oliveira@uib.no

Abstract

Let CMSO denote the counting monadic second order logic of graphs. We give a constructive proof that for some computable function f, there is an algorithm $\mathfrak{A}$ that takes as input a CMSO sentence φ, a positive integer t, and a connected graph G of maximum degree at most Δ, and determines, in time $f(|\varphi|,t)\cdot 2^{O(\Delta\cdot t)}\cdot |G|^{O(t)}$, whether G has a supergraph G' of treewidth at most t such that $G'\models\varphi$.

The algorithmic metatheorem described above sheds new light on certain unresolved questions within the framework of graph completion algorithms. In particular, using this metatheorem, we provide an explicit algorithm that determines, in time $f(d)\cdot 2^{O(\Delta\cdot d)}\cdot |G|^{O(d)}$, whether a connected graph of maximum degree Δ has a planar supergraph of diameter at most d. Additionally, we show that for each fixed k, the problem of determining whether G has an k-outerplanar supergraph of diameter at most d is strongly uniformly fixed parameter tractable with respect to the parameter d.

This result can be generalized in two directions. First, the diameter parameter can be replaced by any contraction-closed effectively CMSO-definable parameter $\mathbf{p}$. Examples of such parameters are vertex-cover number, dominating number, and many other contraction-bidimensional parameters. In the second direction, the planarity requirement can be relaxed to bounded genus, and more generally, to bounded local treewidth.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity, F.4 Mathematical Logic and Formal Languages

Keywords and phrases CMSO Logic, Algorithmic Metatheorems, Graph Completion, Bidimensionality

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.33

1 Introduction

A parameterized problem $\mathcal{L}\subseteq\Sigma^*\times\mathbb{N}$ is said to be *fixed parameter tractable* (FPT) if there exists a function $f:\mathbb{N}\to\mathbb{N}$ such that for each $(x,k)\in\Sigma^*\times\mathbb{N}$, one can decide whether $(x,k)\in\mathcal{L}$ in time $f(k)\cdot|x|^{O(1)}$, where $|x|$ is the size of x [10]. Using non-constructive methods derived from Robertson and Seymour's graph minor theory, one can show that certain problems can be solved in time $f(k)\cdot|x|^{O(1)}$ for some function $f:\mathbb{N}\to\mathbb{N}$. The caveat is that the function f arising from these non-constructive methods is often *not known to be computable*. Interestingly, for some problems it is not even clear how to obtain algorithms running in time $f_1(k)\cdot|x|^{f_2(k)}$ for some *computable* functions f_1 and f_2. In this work we will use techniques from automata theory and structural graph theory to provide constructive FPT and XP algorithms for problems for which only non-constructive parameterized algorithms were known.

* This work was supported by the Bergen Research Foundation.

The counting monadic second-order logic of graphs (CMSO) extends first order logic by allowing quantifications over sets of vertices and sets of edges, and by introducing the notion of modular counting predicates. This logic is expressive enough to define several interesting graph properties, such as Hamiltonicity, 3-colorability, connectivity, planarity, fixed genus, minor embeddability, etc. Additionally, when restricted to graphs of constant treewidth, CMSO logic is able to define precisely those properties that are recognizable by finite state tree-automata operating on encodings of tree-decompositions, or equivalently, those properties that can be described by equivalence relations with finite index [7, 1, 3, 4].

The expressiveness of CMSO logic has had a great impact in algorithmic theory due to Courcelle's model-checking theorem [7]. This theorem states that for some computable function $f : \mathbb{N}^2 \to \mathbb{N}$, one can determine in time[1] $f(|\varphi|, t) \cdot |G|$ whether a given graph G of treewidth at most t satisfies a given CMSO sentence φ. As a consequence of Courcelle's theorem, many combinatorial problems, such as Hamiltonicity or 3-colorability, which are NP-hard on general graphs, can be solved in linear time on graphs of constant treewidth. In this work we will consider a class problems on graphs of constant treewidth which cannot be directly addressed via Courcelle's theorem, either because it is not clear how to formulate the set of positive instances of such a problem as a CMSO-definable set, or because although the set of positive instances is CMSO-definable, it is not clear how to explicitly construct a CMSO sentence φ defining such set. For instance, sets of graphs that are closed under minors very often fall in the second category due to Robertson and Seymour's graph minor theorem.

1.1 Main Result

Let φ be a CMSO sentence, and t be a positive integer. We say that a graph G' is a (φ, t)-supergraph of a graph G if the following conditions are satisfied: G' satisfies φ, G' has treewidth at most t, and G' is a supergraph of G (possibly containing more vertices than G).

In our main result, Theorem 14, we devise an algorithm $\mathfrak{A}$ that takes as input a CMSO sentence φ, a positive integer t, and a connected graph G of maximum degree Δ, and determines in time $f(|\varphi|, t) \cdot 2^{O(\Delta \cdot t)} \cdot |G|^{O(t)}$ whether G has a (φ, t)-supergraph. We note that our algorithm determines the existence of such a (φ, t)-supergraph G' without the need of necessarily constructing G'. Therefore, no bound on the size of a candidate supergraph G' is imposed.

In the next three sub-sections we show how Theorem 14 can be used to provide partial solutions to certain long-standing open problems in parameterized complexity theory.

1.2 Planar Diameter Improvement

In the PLANAR DIAMETER IMPROVEMENT problem (PDI), we are given a graph G, and a positive integer d, and the goal is to determine whether G has a planar supergraph G' of diameter at most d. Note that the set of YES instances for the PDI problem is closed under minors. In other words, if G has a planar supergraph of diameter at most d, then any minor H of G also has such a supergraph. Therefore, using non-constructive arguments from Robertson and Seymour's graph minor theory [17, 18] in conjunction with the fact planar graphs of constant diameter have constant treewidth, one can show that for each fixed d, there exists an algorithm $\mathfrak{A}_d$ which determines in linear time whether a given G has diameter

[1] $|G|$ denotes the number of vertices in G, and $|\varphi|$, the number of symbols in φ.

at most d. The problem is that the non-constructive techniques mentioned above provide us with no clue about what the algorithm $\mathfrak{A}_d$ actually is. This problem can be partially remedied using a technique called effectivization by self-reduction introduced by Fellows and Langston [13, 10]. Using this technique one can show that for some function $f : \mathbb{N} \to \mathbb{N}$, there exists a single algorithm $\mathfrak{A}$ which takes a graph G and a positive integer d as input, and determines in time $f(d) \cdot |G|^{O(1)}$ whether G has a planar supergraph of diameter at most d. The caveat is that the function $f : \mathbb{N} \to \mathbb{N}$ bounding the influence of the parameter d in the running time of the algorithm mentioned above is not known to be computable.

Obtaining a fixed parameter tractability result for the PDI problem with a *computable* function f is a notorious and long-standing open problem in parameterized complexity theory [10, 12, 9]. Indeed, when it comes to explicit algorithms, the status of the PDI problem is much more elusive. As remarked in [5], even the problem of determining whether PDI can be solved in time $f_1(d) \cdot |G|^{f_2(d)}$ for *computable* functions $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ is open.

Using Theorem 14 we provide an explicit algorithm that solves the PDI problem for connected graphs in time $f(d) \cdot 2^{O(\Delta \cdot d)} \cdot |G|^{O(d)}$ where $f : \mathbb{N} \to \mathbb{N}$ is a *computable* function, and Δ is the maximum degree of G. This result settles an open problem stated in [5] in the case in which the input graph is connected and has bounded (even logarithmic) degree. We note that our algorithm imposes no restriction on the degree of a prospective supergraph G'.

1.3 k-Outerplanar Diameter Improvement

A graph is 1-outerplanar if it can be embedded in the plane in such a way that all vertices lie in the outer-face of the embedding. A graph is k-outerplanar if it can be embedded in the plane in such a way that that deleting all vertices in the outer-face of the embedding yields a $(k-1)$-outerplanar graph. The k-outerplanar diameter improvement problem (k-OPDI) is the straightforward variant of PDI in which the completion is required to be k-outerplanar instead of planar. In [5] Cohen at al. provided an explicit polynomial time algorithm for the 1-OPDI problem. The complexity of the k-outerplanar diameter improvement problem was left open for $k \geq 2$. Using Theorem 14 we show that the k-OPDI problem can be solved in time $f(k, d) \cdot 2^{O(\Delta \cdot k)} \cdot |G|^{O(k)}$ where $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a computable function. In other words, for each fixed k, the k-outerplanar diameter improvement problem is strongly uniformly fixed parameter tractable with respect to the diameter parameter d for bounded degree connected input graphs.

1.4 Contraction-Closed Parameters

A graph parameter is a function $\mathbf{p}$ that associates a non-negative integer with each graph. We say that such a parameter is contraction-closed if $\mathbf{p}(G) \leq \mathbf{p}(G')$ whenever G is a contraction of G'. For instance, the diameter of a graph is clearly a contraction-closed parameter. We say that a graph parameter $\mathbf{p}$ is effectively CMSO-definable if there exists a computable function α, and an algorithm that takes a positive integer k as input and constructs a CMSO formula φ_k that is true on a graph G if and only if $\mathbf{p}(G) \leq k$.

The results described in the previous subsections can be generalized in two directions. First, the diameter parameter can be replaced by any effectively CMSO-definable contraction closed parameter that is unbounded on Gamma graphs. These graphs were defined in [15] with the goal to provide a simplified exposition of the theory of contraction-bidimensionality. In particular, many well studied parameters that arise often in bidimensionality theory satisfy the conditions listed above. Examples of such parameters are the sizes of a minimum vertex cover, feedback vertex set, maximal matching, dominating set, edge dominating set,

connected dominating set etc. On the other direction, the planarity requirement can be relaxed to CMSO definable graph properties that exclude some appex graph as a minor. These properties are also known in the literature as bounded local-treewidth properties. For instance, embeddability on surfaces of genus g, for fixed g, is one of such properties.

1.5 Proof Sketch And Organization of the Paper

In Section 2 we state some preliminary definitions. In Section 3 we define the notions of concrete bags, and concrete tree decompositions. Intuitively, a concrete tree-decomposition is an algebraic structure that represents a graph together with one of its tree decompositions. Using such structures we are able to define infinite families of graphs via tree-automata that accept infinite sets of tree decompositions. In particular, Courcelle's theorem can be transposed to this setting. More precisely, there is a computable function f such that for each CMSO sentence φ and each $t \in \mathbb{N}$, one can construct in time $f(|\varphi|, t)$ a tree automaton $\mathcal{A}(\varphi, t)$ which accept precisely those concrete tree decompositions of width at most t that give rise to graphs satisfying φ (Theorem 4).

In Section 4 we define the notion of sub-decomposition of a concrete tree decomposition. Intuitively, if a concrete tree decomposition $\mathbf{T}$ represents a graph G, then a sub-decomposition of $\mathbf{T}$ represents a sub-graph of G. We show that given a tree-automaton $\mathcal{A}$ accepting a set $\mathcal{L}(\mathcal{A})$ of concrete tree decompositions, one can construct a tree automaton $\mathrm{Sub}(\mathcal{A})$ which accepts precisely those sub-decompositions of concrete tree decompositions in $\mathcal{L}(\mathcal{A})$ (Theorem 6).

In Section 5, we introduce the main technical tool of this work. More specifically, we show that for each connected graph G of maximum degree Δ, one an construct in time $2^{O(\Delta \cdot t)} \cdot |G|^{O(t)}$ a tree-automaton $\mathcal{A}(G, t)$ whose language $\mathcal{L}(\mathcal{A}(G, t))$ consists precisely of those concrete tree decompositions of width at most t that give rise to G (Theorem 12).

In Section 6 we argue that the problem of determining whether G has a supergraph of treewidth at most t satisfying φ is equivalent to determining whether the intersection of $\mathcal{L}(\mathcal{A}(G, t+1))$ with $\mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1)))$ is non-empty. By combining Theorems 4, 6 and 12, we infer that this problem can be solved in time $f(|\varphi|, t) \cdot 2^{O(\Delta \cdot t)} \cdot |G|^{O(t)}$ (Theorem 14). Finally, in Section 7, we apply Theorem 14 to obtain explicit algorithms for several supergraph problems involving contraction-closed parameters.

2 Preliminaries

For each $n \in \mathbb{N}$, we let $[n] = \{1, ..., n\}$. We let $[0] = \emptyset$. For each finite set U, we let $\mathcal{P}(U)$ denote the set of subsets of U. For each $r \in \mathbb{N}$ and each finite set U, we let $\mathcal{P}^{\leq}(U, r) = \{U' \subseteq U \mid |U'| \leq r\}$ be the set of subsets of U of size at most r, and $\mathcal{P}^{=}(U, r) = \{U' \subseteq U \mid |U'| = r\}$ be the set of subsets of X of size precisely r. If $A, A_1, ..., A_k$ are sets, then we write $A = A_1 \dot\cup A_2 \dot\cup ... \dot\cup A_k$ to indicate that $A_i \cap A_j = \emptyset$ for $i \neq j$, and that A is the disjoint union of $A_1, .., A_k$.

Graphs. A *graph* is a triple $G = (V_G, E_G, \mathrm{Inc}_G)$ where V_G is a set of vertices, E_G is a set of edges, and $\mathrm{Inc}_G \subseteq E_G \times V_G$ is an incidence relation. For each $e \in E_G$ we let $\mathit{endpts}(e) = \{v \mid \mathrm{Inc}_G(e, v)\}$ be the set of endpoints of e, and we assume that $|\mathit{endpts}(e)|$ is either 0 or 2. We note that our graphs are allowed to have multiple edges, but no loops. We say that a graph H is a subgraph of G if $V_H \subseteq V_G$, $E_H \subseteq E_G$ and $\mathrm{Inc}_H = \mathrm{Inc}_G \cap E_H \times V_H$. Alternatively, we say that G is a supergraph of H. The degree of a vertex $v \in V_G$ is the number $d(v)$ of edges incident with v. We let $\Delta(G)$ denote the maximum degree of a vertex of G.

A *path* in a graph G is a sequence $v_1e_1v_2...e_{n-1}v_n$ where $v_i \in V_G$ for $i \in [n]$, $e_i \in E_G$ for $i \in [n-1]$, $v_i \neq v_j$ for $i \neq j$, and $\{v_i, v_{i+1}\} = \mathit{endpts}(e_i)$ for each $i \in [n-1]$. We say that G is *connected* if for every two vertices $v, v' \in V_G$ there is a path whose first vertex is v and whose last vertex is v'.

Let G and H be graphs. An isomorphism from G to H is a pair of bijections $\mu = (\dot{\mu} : V_G \to V_H, \overline{\mu} : E_G \to E_H)$ such that for every $e \in E_G$ if $\mathit{endpts}(e) = \{v, v'\}$ then $\mathit{endpts}(\overline{\mu}(e)) = \{\dot{\mu}(v), \dot{\mu}(v')\}$. We say that G and H are isomorphic if there is an isomorphism from G to H.

Treewidth. A tree is an acyclic graph T containing a unique connected component. To avoid confusion we may call the vertices of a tree "nodes" and call their edges "arcs". We let $\mathit{nodes}(T)$ denote the set of nodes of T and $\mathit{arcs}(T)$ denote its set of arcs. A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : \mathit{nodes}(T) \to \mathcal{P}(V_G)$ is a function that labels nodes of T with subsets of vertices of G in such a way that the following conditions are satisfied.

1. $\bigcup_{u \in \mathit{nodes}(T)} \beta(u) = V_G$
2. For every $e \in E_G$, there exists a node $u \in \mathit{nodes}(T)$ such that $\mathit{endpts}(e) \subseteq \beta(u)$
3. For every $v \in V_G$, the set $T_v = \{u \in \mathit{nodes}(T) \mid v \in \beta(u)\}$, i.e., the set of nodes of T whose corresponding bags contain v, induces a connected subtree of T.

The *width* of a tree decomposition (T, β) is defined as $\mathit{max}_{u \in \mathit{nodes}(T)} |\beta(u)| - 1$, that is, the maximum bag size minus one. The treewidth of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width of a tree decomposition of G.

CMSO Logic. The counting monadic second-order logic of graphs, here denoted by CMSO, extends first order logic by allowing quantifications over sets of vertices and edges, and by introducing the notion of modular counting predicates. More precisely, the syntax of CMSO logic includes the logical connectives $\vee, \wedge, \neg, \Leftrightarrow, \Rightarrow$, variables for vertices, edges, sets of vertices and sets of edges, the quantifiers $\exists, \forall$ that can be applied to these variables, and the following atomic predicates:

1. $x \in X$ where x is a vertex variable and X a vertex-set variable;
2. $y \in Y$ where y is an edge variable and Y an edge-set variable;
3. $\mathrm{Inc}(x, y)$ where x is a vertex variable, y is an edge variable, and the interpretation is that the edge x is incident with the edge y.
4. $\mathit{card}_{s,r}(Z)$ where $0 \leq s < r$, $r \geq 2$, Z is a vertex-set or edge-set variable, and the interpretation is that $|Z| = s \pmod r$;
5. equality of variables representing vertices, edges, sets of vertices and sets of edges.

A CMSO *sentence* is a CMSO formula without free variables. If φ is a CMSO sentence, then we write $G \models \varphi$ to indicate that G satisfies φ.

Terms. Let Σ be a finite set. The set $\mathit{Ter}(\Sigma)$ of terms over Σ is inductively defined as follows.

1. If $a \in \Sigma$, then $a \in \mathit{Ter}(\Sigma)$.
2. If $a \in \Sigma$, and $t \in \mathit{Ter}(\Sigma)$, then $a(t) \in \mathit{Ter}(\Sigma)$.
3. If $a \in \Sigma$, and $t_1, t_2 \in \mathit{Ter}(\Sigma)$, then $a(t_1, t_2) \in \mathit{Ter}(\Sigma)$.

Note that the alphabet Σ is unranked and the symbols in Σ may be regarded as function symbols or arity 0, 1 or 2. The set of positions of a term $t = a(t_1, ..., t_r) \in \mathit{Ter}(\Sigma)$ is defined

as follows.

$$Pos(t) = \{\lambda\} \cup \bigcup_{j \in \{1,\dots,r\}} \{jp \mid p \in Pos(t_j)\}.$$

Note that if $t = a$ for some $a \in \Sigma$, then $Pos(t) = \{\lambda\}$. If $p, pj \in Pos(t)$ where $j \in \{1,2\}$, then we say that pj is a *child* of p. Alternatively, we say that p is the *parent* of pj. We say that p is a *leaf* if it has no children. We let $\tau(t)$ be the tree that has $Pos(t)$ as nodes and $\{\{p, pj\} \mid j \in \{1,2\},\ p, pj \in Pos(t)\}$ as arcs. We say that a subset $P \subseteq Pos(t)$ is *connected* if the sub-tree of $\tau(t)$ induced by P is connected. If P is connected, then we say that a position $p \in P$ is the root of P if the parent of p does not belong to P.

If $t = a(t_1, ..., t_r)$ is a term in $Ter(\Sigma)$ for $r \in \{0,1,2\}$, and $p \in Pos(t)$, then the symbol $t[p]$ at position p is inductively defined as follows. If $p = \lambda$, then $t[p] = a$. On the other hand, if $p = jp'$ where $j \in \{1,2\}$, then $t[p] = t_j[p']$.

Tree Automata. Let Σ be a finite set of symbols. A *bottom-up tree-automaton* over Σ is a tuple $\mathcal{A} = (Q, \Sigma, F, \Delta)$ where Q is a set of states, $F \subseteq Q$ a set of final states, and Δ is a set of transitions of the form $(\mathfrak{q}_1, ..., \mathfrak{q}_r, a, \mathfrak{q})$ with $a \in \Sigma$, $0 \leq r \leq 2$, and $\mathfrak{q}_1, ..., \mathfrak{q}_r, \mathfrak{q} \in Q$. The size of $\mathcal{A}$, which is defined as $|\mathcal{A}| = |Q| + |\Delta|$, measures the number of states in Q plus the number of transitions in Δ. The set $\mathcal{L}(\mathcal{A}, \mathfrak{q}, i)$ of all terms reaching a state $\mathfrak{q} \in Q$ in depth at most i is inductively defined as follows.

$$\mathcal{L}(\mathcal{A}, \mathfrak{q}, 1) = \{a \mid (a, \mathfrak{q}) \in \Delta\}$$

$$\mathcal{L}(\mathcal{A}, \mathfrak{q}, i) = \mathcal{L}(\mathcal{A}, \mathfrak{q}, i-1) \cup \{a(t_1, ..., t_r) \mid r \in \{1,2\}, \text{ and } \exists(\mathfrak{q}_1, ..., \mathfrak{q}_r, a, \mathfrak{q}) \in \Delta,\ t_j \in \mathcal{L}(\mathcal{A}, \mathfrak{q}_j, i-1)\}$$

We denote by $\mathcal{L}(\mathcal{A}, \mathfrak{q})$ the set of all terms reaching state $\mathfrak{q}$ in finite depth, and by $\mathcal{L}(\mathcal{A})$ the set of all terms reaching some final state in F.

$$\mathcal{L}(\mathcal{A}, \mathfrak{q}) = \bigcup_{i \in \mathbb{N}} \mathcal{L}(\mathcal{A}, \mathfrak{q}, i) \qquad \mathcal{L}(\mathcal{A}) = \bigcup_{\mathfrak{q} \in F} \mathcal{L}(\mathcal{A}, \mathfrak{q}) \tag{1}$$

We say that the set $\mathcal{L}(\mathcal{A})$ is the language *accepted* by $\mathcal{A}$.

Let $\boldsymbol{\pi} : \Sigma \to \Sigma'$ be a map between finite sets of symbols Σ and Σ'. Such mapping can be homomorphically extended to a mapping $\boldsymbol{\pi} : Ter(\Sigma) \to Ter(\Sigma')$ between terms by setting $\boldsymbol{\pi}(t)[p] = \boldsymbol{\pi}(t[p])$ for each position $p \in Pos(t)$. Additionally, $\boldsymbol{\pi}$ can be further extended to a set of terms $\mathcal{L} \subseteq Ter(\Sigma)$ by setting $\boldsymbol{\pi}(\mathcal{L}) = \{\boldsymbol{\pi}(t) \mid t \in Ter(\Sigma)\}$. Below we state some well known closure and decidability properties for tree automata.

▶ **Lemma 1** (Properties of Tree Automata [6]). *Let Σ and Σ' be finite sets of symbols. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be tree automata over Σ, and $\boldsymbol{\pi} : \Sigma \to \Sigma'$ be a mapping.*

1. *One can construct in time $O(|\mathcal{A}_1| \cdot |\mathcal{A}_2|)$ a tree automaton $\mathcal{A}_1 \cap \mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_1 \cap \mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.*
2. *One can determine whether $\mathcal{L}(\mathcal{A}_1) \neq \emptyset$ in time $|\mathcal{A}_1|^{O(1)}$.*
3. *One can construct in time $O(|\mathcal{A}_1|)$ a tree automaton $\boldsymbol{\pi}(\mathcal{A}_1)$ such that $\mathcal{L}(\boldsymbol{\pi}(\mathcal{A}_1)) = \boldsymbol{\pi}(\mathcal{L}(\mathcal{A}_1))$.*

3 Concrete Tree Decompositions

A *t-concrete bag* is a pair (B, b) where $B \subseteq [t]$, and $b \subseteq B$ with $b = \emptyset$ or $|b| = 2$. We note that B is allowed to be empty. We let $\mathcal{B}(t)$ be the set of all t-concrete bags. Note that

$|\mathcal{B}(t)| \leq t^2 \cdot 2^t$. We regard the set $\mathcal{B}(t)$ as a finite alphabet which will be used to construct terms representing tree decompositions of graphs.

A t-concrete tree decomposition is a term $\mathbf{T} \in \mathit{Ter}(\mathcal{B}(t))$. We let $\mathbf{T}[p] = (\mathbf{T}[p].B, \mathbf{T}[p].b)$ be the t-concrete bag at position p of $\mathbf{T}$. For each $s \in [t]$, we say that a subset $P \subseteq \mathit{Pos}(\mathbf{T})$ is *s-maximal* if the following conditions are satisfied.

1. P is connected in $\mathit{Pos}(\mathbf{T})$.
2. $s \in \mathbf{T}[p].B$ for every $p \in P$.
3. If P' is a connected subset of $\mathit{Pos}(\mathbf{T})$ and $s \in \mathbf{T}[p].B$ for every $s \in P'$, then $P' \subseteq P$.

Note that if P and P' are s-maximal then either $P = P'$, or $P \cap P' = \emptyset$. Additionally, $p \in \mathit{Pos}(\mathbf{T})$ and each $s \in \mathbf{T}[p].B$, there exists a unique subset $P \subseteq \mathit{Pos}(\mathbf{T})$ such that P is s-maximal and $p \in P$. We denote this unique set by $P(p, s)$.

▶ **Definition 2.** Let $\mathbf{T} \in \mathit{Ter}(\mathcal{B}(t))$. The graph $\mathcal{G}(\mathbf{T})$ associated with $\mathbf{T}$ is defined as follows.

1. $V_{\mathcal{G}(\mathbf{T})} = \{v_{s,P} \mid s \in [t],\ P \subseteq \mathit{Pos}(\mathbf{T}),\ P \text{ is } s\text{-maximal}\}$.
2. $E_{\mathcal{G}(\mathbf{T})} = \{e_p \mid p \in \mathit{Pos}(\mathbf{T}),\ b \neq \emptyset\}$.
3. $\mathrm{Inc}_{\mathcal{G}(\mathbf{T})} = \{(e_p, v_{s,P(p,s)}) \mid e_p \in E_{\mathcal{G}(\mathbf{T})},\ s \in \mathbf{T}[p].b\}$.

Intuitively, a t-concrete tree decomposition may be regarded as a way of representing a graph together with one of its tree decompositions. This idea is widespread in texts dealing with recognizable properties of graphs [3, 2, 8, 11, 14]. Within this framework it is customary to define a bag of width t as a graph with at most t vertices together with a function that labels the vertices of these graphs with numbers from $\{1, ..., t\}$. Our notion of t-concrete bag, on the other hand, may be regarded as a representation of a graph with at most t vertices injectively labeled with numbers from $\{1, ...t\}$ and at most *one* edge. Within this point of view, the representation used here is a syntactic restriction of the former. On the other hand, any decomposition which uses bags with arbitrary graphs of size t can be converted into a t-concrete decomposition, by expanding each bag into a sequence of t^2 concrete bags. The following observation is immediate, using the fact that if a graph has treewidth t, then it has a rooted tree decomposition in which each node has at most two children [11].

▶ **Observation 3.** *A graph G has treewidth t if and only if there exists some $(t+1)$-concrete tree decomposition $\mathbf{T} \in \mathit{Ter}(\mathcal{B}(t+1))$ such that $\mathcal{G}(\mathbf{T})$ is isomorphic to G.*

The next theorem may be regarded as a variant of Courcelle's theorem [8].

▶ **Theorem 4** (Courcelle's Theorem)**.** *There exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for each CMSO sentence φ, and each $t \in \mathbb{N}$, there is a tree-automaton $\mathcal{A}(\varphi, t)$ accepting the following tree language.*

$$\mathcal{L}(\mathcal{A}(\varphi, t)) = \{\mathbf{T} \in \mathit{Ter}(\mathcal{B}(t)) \mid \mathcal{G}(\mathbf{T}) \models \varphi\}. \tag{2}$$

4 Sub-Decompositions

In this section we introduce the notion of sub-decompositions of a t-concrete decomposition. Intuitively, if a t-concrete tree decomposition $\mathbf{T}$ represents a graph G then sub-decompositions of $\mathbf{T}$ represent subgraphs of G. The main result of this section states that given a tree automaton $\mathcal{A}$ over $\mathcal{B}(t)$, one can efficiently construct a tree automaton $\mathrm{Sub}(\mathcal{A})$ over $\mathcal{B}(t)$ which accepts precisely those sub-decompositions of t-concrete tree decompositions in $\mathcal{L}(\mathcal{A})$.

We say that a t-concrete bag (B, b) is a sub-bag of a t-concrete bag (B', b') if $B \subseteq B'$ and $b \subseteq b'$.

▶ **Definition 5.** We say that a t-concrete tree decomposition $\mathbf{T} \in Ter(\mathcal{B}(t))$ is a sub-decomposition of a t-concrete tree decomposition $\mathbf{T}' \in Ter(\mathcal{B}(t))$ if the following conditions are satisfied.

S1. $Pos(\mathbf{T}) = Pos(\mathbf{T}')$.

S2. For each $p \in Pos(\mathbf{T})$, $\mathbf{T}[p]$ is a sub-bag of $\mathbf{T}'[p]$.

S3. For each $p, pj \in Pos(\mathbf{T})$, and for each $s \in [t]$, if $s \in \mathbf{T}'[p].B$ and $s \in \mathbf{T}'[pj].B$, then $s \notin \mathbf{T}[p].B$ if and only if $s \notin \mathbf{T}[pj].B$.

The following theorem states that sub-decompositions of $\mathbf{T}$ are in one to one correspondence with subgraphs of $\mathcal{G}(\mathbf{T})$.

▶ **Theorem 6.** *Let G and G' be a graphs and let $\mathbf{T}' \in Ter(\mathcal{B}(t))$ be a t-concrete tree decomposition of G'. Then G is a subgraph of G' if and only if there exists some $\mathbf{T} \in Ter(\mathcal{B}(t))$ such that $\mathbf{T}$ is a sub-decomposition of $\mathbf{T}'$ with $\mathcal{G}(\mathbf{T}) = G$.*

Proof.

1. Let G be a subgraph of $\mathcal{G}(\mathbf{T}')$. We show that there exists a sub-decomposition $\mathbf{T}$ of $\mathbf{T}'$ such that $\mathcal{G}(\mathbf{T}) = G$. Since G is a subgraph of $\mathcal{G}(\mathbf{T})$, we have that $V_G \subseteq V_{\mathcal{G}(\mathbf{T}')}$, $E_G \subseteq E_{\mathcal{G}(\mathbf{T}')}$, and $\mathrm{Inc}_G = \mathrm{Inc}_{\mathcal{G}(\mathbf{T}')} \cap E_G \times V_G$. We define $\mathbf{T}$ by setting $\mathbf{T}[p]$ as follows for each $p \in Pos(\mathbf{T}) = Pos(\mathbf{T}')$.
   a. $\mathbf{T}[p].B = \mathbf{T}'[p].B \backslash \{s \mid v_{s,P(p,s)} \in V_{\mathcal{G}(\mathbf{T}')} \backslash V_G\}$.
   b. $\mathbf{T}[p].b = \emptyset$ if $e_p \in E_{\mathcal{G}(\mathbf{T}')} \backslash E_G$ and $\mathbf{T}[p].b = \mathbf{T}'[p].b$ otherwise.
   First, we note that $v_{s,P} \in V_{\mathcal{G}(\mathbf{T})}$ if and only if $v_{s,P} \in V_G$, $e_p \in E_{\mathcal{G}(\mathbf{T})}$ if and only if $e_p \in E_G$, and $(e_p, v_{i,P}) \in \mathrm{Inc}_{\mathcal{G}(\mathbf{T})}$ if and only if $(e_p, v_{i,P}) \in V_G$. Therefore, $G = \mathcal{G}(\mathbf{T})$. We will check that the t-concrete decomposition $\mathbf{T}$ defined above is indeed a sub-decomposition of $\mathbf{T}'$. In other words, we will verify that conditions **S1**, **S2** and **S3** above are satisfied. The fact that **S1** is satisfied is immediate, since we define $\mathbf{T}[p]$ for each $p \in Pos(\mathbf{T}')$. Therefore, $Pos(\mathbf{T}) = Pos(\mathbf{T}')$. Condition **S2** is also satisfied, since by (a) and (b) we have that $\mathbf{T}[p].B \subseteq \mathbf{T}'[p].B$ and that $\mathbf{T}[p].b$ is either $\emptyset$, or equal to $\mathbf{T}'[p].b$. Finally, condition **S3** is also satisfied, since (a) guarantees that for each number $s \in [t]$, and each s-maximal set $P \subseteq Pos(\mathbf{T}')$, if s is removed from $\mathbf{T}'[p].B$ for some $p \in P$, then s is indeed removed from $\mathbf{T}'[p].B$ for every $p \in P$.
2. For the converse, let $\mathbf{T}$ be a sub-decomposition of $\mathbf{T}'$. We show that the graph $\mathcal{G}(\mathbf{T})$ is a subgraph of $\mathcal{G}(\mathbf{T}')$. First, we note that condition **S3** guarantees that for each $s \in [t]$ and each $P \subseteq Pos(\mathbf{T})$, if P is s-maximal in $\mathbf{T}$ then P is s-maximal in $\mathbf{T}'$. Therefore, $V_{\mathcal{G}(\mathbf{T})} \subseteq V_{\mathcal{G}(\mathbf{T}')}$. Now, Condition **S2** guarantees that $e_p \in E_{\mathcal{G}(\mathbf{T})}$ implies that $e_p \in E_{\mathcal{G}(\mathbf{T}')}$. Therefore, $E_{\mathcal{G}(\mathbf{T})} \subseteq E_{\mathcal{G}(\mathbf{T}')}$. Finally, by definition $(e_p, v_{s,P}) \in \mathrm{Inc}_{\mathcal{G}(\mathbf{T})}$ if and only if $s \in \mathbf{T}[p].b$ for each $p \in P$. Since the fact that $s \in \mathbf{T}[p].b$ implies that $s \in \mathbf{T}'[p].b$, we have that $(e_p, v_{s,P}) \in \mathrm{Inc}_{\mathcal{G}(\mathbf{T})}$ implies that $(e_p, v_{s,P}) \in \mathrm{Inc}_{\mathcal{G}(\mathbf{T}')}$. Therefore, $\mathrm{Inc}_{\mathcal{G}(\mathbf{T})} \subseteq \mathrm{Inc}_{\mathcal{G}(\mathbf{T}')}$. Additionally, since $(e_p, v_{s,P(s,p)}) \in \mathrm{Inc}_{\mathcal{G}(\mathbf{T})}$ for each $e_p \in E_{\mathcal{G}(\mathbf{T})}$ and each $s \in \mathbf{T}[p].b$, we have that $\mathrm{Inc}_{\mathcal{G}(\mathbf{T})} = \mathrm{Inc}_{\mathcal{G}(\mathbf{T}')} \cap E_{\mathcal{G}(\mathbf{T})} \times V_{\mathcal{G}(\mathbf{T})}$. This shows that $\mathcal{G}(\mathbf{T})$ is a subgraph of $\mathcal{G}(\mathbf{T}')$. ◀

The following theorem states that given a tree automaton $\mathcal{A}$ over $\mathcal{B}(t)$, one can efficiently construct a tree automaton $\mathrm{Sub}(\mathcal{A})$ which accepts precisely those sub-decompositions of t-concrete tree decompositions in $\mathcal{L}(\mathcal{A})$.

▶ **Theorem 7** (Sub-Decomposition Automaton)**.** *Let $\mathcal{A}$ be a tree automaton over $\mathcal{B}(t)$. Then one can construct in time $2^{O(t)} \cdot |\mathcal{A}|$ a tree automaton $\mathrm{Sub}(\mathcal{A})$ over $\mathcal{B}(t)$ accepting the following language.*

$$\mathcal{L}(\mathrm{Sub}(\mathcal{A})) = \{\mathbf{T} \in Ter(\mathcal{B}(t)) \mid \exists \mathbf{T}' \in \mathcal{L}(\mathcal{A}) \text{ s.t. } \mathbf{T} \text{ is a sub-decomposition of } \mathbf{T}'\}.$$

Proof. Let $\mathcal{A} = (Q, \mathcal{B}(t), \Delta, F)$ be a tree automaton over $\mathcal{B}(t)$. As a first step we create an intermediate tree automaton $\mathcal{A}' = (Q', \mathcal{B}(t), \Delta', F')$ which accepts the same language as $\mathcal{A}$. The tree automaton $\mathcal{A}'$ is defined as follows.

$$Q' = \{\mathfrak{q}_B \mid \mathfrak{q} \in Q,\ B \subseteq [t]\} \qquad F' = \{\mathfrak{q}_B \mid \mathfrak{q} \in F,\ B \subseteq [t]\}$$

$$\Delta' = \{(\mathfrak{q}^1_{B_1}, ..., \mathfrak{q}^r_{B_r}, (B, b), \mathfrak{q}_B) \mid (\mathfrak{q}^1, ..., \mathfrak{q}^r, (B, b), \mathfrak{q}) \in \Delta,\ B_i \subseteq [t] \text{ for } i \in [r]\}.$$

Note that for each $\mathfrak{q} \in Q$, each $B \subseteq [t]$, and each $\mathbf{T} \in \mathit{Ter}(\mathcal{B}(t))$, $\mathbf{T}$ reaches state $\mathfrak{q}_B$ in $\mathcal{A}'$ if and only if $\mathbf{T}$ reaches state $\mathfrak{q}$ in $\mathcal{A}$ and $\mathbf{T}[\lambda].B = B$, where $\mathbf{T}[\lambda]$ is the t-concrete bag at the root of $\mathbf{T}$. In particular, this implies that a term $\mathbf{T}$ belongs to $\mathcal{L}(\mathcal{A}')$ if and only if $\mathbf{T} \in \mathcal{L}(\mathcal{A})$.

Now, consider the tree automaton $\mathrm{Sub}(\mathcal{A}) = (Q'', \mathcal{B}(t), \Delta'', F'')$ over $\mathcal{B}(t)$ where

$$Q'' = \{\mathfrak{q}_{B,B'} \mid \mathfrak{q} \in Q, B \subseteq B' \subseteq [t]\} \qquad F'' = \{\mathfrak{q}_{B,B'} \mid \mathfrak{q} \in F, B \subseteq B' \subseteq [t]\}$$

$$\begin{aligned}\Delta'' = \{(\mathfrak{q}^1_{B_1,B'_1}, ..., \mathfrak{q}^r_{B_r,B'_r}, (B, b), \mathfrak{q}_{B,B'}) \mid\ & \exists (\mathfrak{q}^1_{B'_1}, ..., \mathfrak{q}^r_{B'_r}, (B', b'), \mathfrak{q}_{B'}) \in \Delta' \text{ such that} \\ & B_i \subseteq B'_i,\ B \subseteq B', \\ & (B, b) \text{ is a sub-bag of } (B', b') \\ & \text{for each } j \in [r], \text{ if } s \in B' \wedge s \in B'_j \text{ then } s \notin B \Leftrightarrow s \notin B_j\}.\end{aligned}$$

It follows by induction on the height of terms that a term $\mathbf{T} \in \mathit{Ter}(\mathcal{B}(t))$ reaches a state $\mathfrak{q}_{B,B'}$ in $\mathrm{Sub}(\mathcal{A})$ if and only if there exists some term $\mathbf{T}' \in \mathit{Ter}(\mathcal{B}(t))$ such that $\mathbf{T}'$ reaches state $\mathfrak{q}_{B'}$ in $\mathcal{A}'$, $\mathbf{T}[\lambda].B = B$, $\mathbf{T}'[\lambda].B = B'$, and $\mathbf{T}$ is a sub-decomposition of $\mathbf{T}'$. In particular, $\mathbf{T}$ reaches a final state of $\mathrm{Sub}(\mathcal{A})$ if and only if $\mathbf{T}$ is a sub-decomposition of some $\mathbf{T}'$ which reaches a final state of $\mathcal{A}'$. ◀

5 Representing All Tree Decompositions of a Given Graph

In this section we show that given a connected graph G of maximum degree Δ, and a positive integer t, one can construct in time $2^{O(\Delta \cdot t)} \cdot |V_G|^{O(t)}$ a tree automaton $\mathcal{A}(G, t)$ over $\mathcal{B}(t)$ that accepts precisely those t-concrete tree decompositions of G.

Let G be a graph. A (G, t)-concrete bag is a tuple $(B, b, \nu, \eta, y, \rho)$ where (B, b) is a t-concrete bag; $\nu : B \to V_G$ is a function that assigns a vertex of G to each element of B; $\eta : B \to \mathcal{P}^{\leq}(E_G, \Delta(G))$ is a function that assigns to each element $s \in B$, a set of edges incident with $\nu(s)$ of size at most $\Delta(G)$; y is a subset of E_G such that $|y| \leq 1$ and $y \subseteq \eta(s)$ whenever $s \in b$; and ρ is a subset of B.

We let $\mathcal{B}(G, t)$ be the set of all (G, t)-concrete bags. Note that $\mathcal{B}(G, t)$ has at most $2^{O(\Delta(G) \cdot t)} \cdot |V_G|^{O(t)}$ elements. We let $\mathit{Ter}(\mathcal{B}(G, t))$ be the set of all terms over $\mathcal{B}(G, t)$. If $\hat{\mathbf{T}}$ is a term in $\mathcal{B}(G, t)$ then for each $p \in \mathit{Pos}(\mathbf{T})$, the (G, t)-concrete bag of $\hat{\mathbf{T}}$ at position p is denoted by the tuple

$$(\hat{\mathbf{T}}[p].B, \hat{\mathbf{T}}[p].b, \hat{\mathbf{T}}[p].\nu, \hat{\mathbf{T}}[p].\eta, \hat{\mathbf{T}}[p].y, \hat{\mathbf{T}}[p].\rho).$$

▶ **Definition 8.** We say that a term $\hat{\mathbf{T}} \in \mathit{Ter}(\mathcal{B}(G, t))$ is a (G, t)-concrete tree decomposition if the following conditions are satisfied for each each $p \in \mathit{Pos}(\hat{\mathbf{T}})$ and each $s \in [t]$.

C1. If $pj \in \mathit{Pos}(\hat{\mathbf{T}})$ and $s \in \hat{\mathbf{T}}[p].B \cap \hat{\mathbf{T}}[pj].B$ then $\hat{\mathbf{T}}[p].\nu(s) = \hat{\mathbf{T}}[pj].\nu(s)$.

C2. If $\hat{\mathbf{T}}[p].b = \{s, s'\}$ then $\hat{\mathbf{T}}[p].y = \{e\}$ for some edge e with

$$\mathit{endpts}(e) = \{\hat{\mathbf{T}}[p].\nu(s), \hat{\mathbf{T}}[p].\nu(s')\}$$

.

C3. Let $r \in \{0,1,2\}$, and $p1, ..., pr$ be the children[2] of p, then

$$\hat{\mathbf{T}}[p].\eta(s) = \hat{\mathbf{T}}[p].y \dot{\cup} \hat{\mathbf{T}}[p1].\eta(s) \dot{\cup} ... \dot{\cup} \hat{\mathbf{T}}[pr].\eta(s).$$

C4. If $s \in \hat{\mathbf{T}}[p].\rho$ then $\hat{\mathbf{T}}[p].\eta(s) = \{e \mid (e, \hat{\mathbf{T}}[p].\nu(s)) \in \mathrm{Inc}_G\}$.

C5. If $p = \lambda$ then $\hat{\mathbf{T}}[p].\rho = \hat{\mathbf{T}}[p].B$. If $pj \in Pos(\hat{\mathbf{T}})$ then $s \in \hat{\mathbf{T}}[pj].\rho$ if and only if $s \in \hat{\mathbf{T}}[pj].B$ and $s \notin \hat{\mathbf{T}}[p].B$.

Let $\boldsymbol{\pi} : \mathcal{B}(G,t) \to \mathcal{B}(t)$ be a function such that $\boldsymbol{\pi}(B, b, \nu, \eta, y, \rho) = (B, b)$ for each (G,t)-concrete bag $(B, b, , \nu, \eta, y, \rho) \in \mathcal{B}(G,t)$. In other words, $\boldsymbol{\pi}$ transforms a (G,t)-concrete bag into a t-concrete bag by erasing the four last coordinates of the former. If $\hat{\mathbf{T}}$ is a term in $Ter(\mathcal{B}(G,t))$ then we let $\boldsymbol{\pi}(\hat{\mathbf{T}})$ be the term in $Ter(\mathcal{B}(t))$ which is obtained by setting $\boldsymbol{\pi}(\hat{\mathbf{T}})[p] = \boldsymbol{\pi}(\hat{\mathbf{T}}[p])$ for each position $p \in Pos(\hat{\mathbf{T}})$.

▶ **Theorem 9.** *Let G be a connected graph and let $\mathbf{T} \in Ter(\mathcal{B}(t))$. Then $\mathbf{T}$ is a t-concrete tree decomposition of G if and only if $|V_{\mathcal{G}(\mathbf{T})}| = |V_G|$ and there exists a (G,t)-concrete tree decomposition $\hat{\mathbf{T}} \in Ter(\mathcal{B}(G,t))$ such that $\mathbf{T} = \boldsymbol{\pi}(\hat{\mathbf{T}})$.*

Note that conditions **C1**-**C5** are local in the sense that they may be verified at each position $p \in Pos(\hat{\mathbf{T}})$ by analysing only the concrete bags $\hat{\mathbf{T}}[p], \hat{\mathbf{T}}[p1], ..., \hat{\mathbf{T}}[pr]$ where $p1, ..., pr$ are the children of p. This allows us to define a tree automaton $\hat{\mathcal{A}}(G,t)$ over $\mathcal{B}(G,t)$ that accepts a term $\hat{\mathbf{T}} \in Ter(\mathcal{B}(G,t))$ if and only if $\hat{\mathbf{T}}$ is a (G,t)-concrete tree decomposition.

▶ **Lemma 10.** *For each positive integer t and each graph G of maximum degree Δ, one can construct in time $2^{O(\Delta)} \cdot |V_G|^{O(t)}$ a tree automaton $\hat{\mathcal{A}}(G,t)$ over $\mathcal{B}(G,t)$ accepting the following language.*

$$\mathcal{L}(\hat{\mathcal{A}}(G,t)) = \{\hat{\mathbf{T}} \in Ter(\mathcal{B}(G,t)) \mid \hat{\mathbf{T}} \text{ is a } (G,t)\text{-concrete tree decomposition.}\} \tag{3}$$

The next lemma states that for each positive integers t and n, one can efficiently construct a tree automaton $\mathcal{A}(t,n)$ which accepts precisely those t-concrete tree decompositions which give rise to graphs with n vertices.

▶ **Lemma 11.** *Let t and n be positive integers with $t \leq n$. One can construct in time $2^{O(t)} \cdot n^3$ a tree automaton $\mathcal{A}(t,n)$ over $\mathcal{B}(t)$ accepting the following language.*

$$\mathcal{L}(\mathcal{A}(t,n)) = \{\mathbf{T} \in Ter(\mathcal{B}(t)) \mid |V_{\mathcal{G}(\mathbf{T})}| = n\}$$

The main result of this section (Theorem 12), follows by a combination of Theorem 9, Lemma 10 and Lemma 11.

▶ **Theorem 12.** *Let G be a connected graph of treewidth t and maximum degree Δ. Then one can construct in time $2^{O(\Delta \cdot t)} \cdot |V_G|^{O(t)}$ a tree automaton $\mathcal{A}(G,t)$ over $\mathcal{B}(t)$ such that for each $\mathbf{T} \in Ter(\mathcal{B}(t))$, $\mathbf{T} \in \mathcal{L}(\mathcal{A}(G,t))$ if and only if $\mathbf{T}$ is a concrete tree decomposition of G.*

6 (φ, t)-Supergraphs

Let φ be a CMSO sentence, and t be a positive integer. Let G and G' be graphs. We say that G' is a (φ, t)-supergraph of G if the following three conditions are satisfied: $G' \models \varphi$, G' has treewidth at most t, and G is a subgraph of G'.

[2] If $r = 0$ then p has no child.

▶ **Lemma 13.** *Let φ be a CMSO sentence and t be a positive integer. Then a graph G has a (φ, t)-supergraph if and only if there exists a $(t+1)$-concrete tree decomposition $\mathbf{T} \in \mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1))$ such that $\mathcal{G}(\mathbf{T})$ is isomorphic to G.*

Proof. Assume that G is a graph that has a (φ, t)-supergraph G'. Then G' satisfies φ, G' has treewidth at most t, and G is a subgraph of G'. By Observation 3, G' has a $(t+1)$-concrete tree decomposition $\mathbf{T}' \in \mathit{Ter}(\mathcal{B}(t+1))$, and therefore by Theorem 4, $\mathbf{T}' \in \mathcal{L}(\mathcal{A}(\varphi, t))$. Since G is a subgraph of G', by Theorem 6, $\mathbf{T}'$ has a sub-decomposition $\mathbf{T}$ which is a $(t+1)$-concrete tree decomposition of G. Therefore, $\mathbf{T}$ belongs to $\mathrm{Sub}(\mathcal{A}(\varphi, t+1))$.

For the converse, let $\mathbf{T} \in \mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1)))$ and let $\mathbf{T}$ be a $(t+1)$-concrete tree decomposition of G. Then $\mathbf{T}$ is a sub-decomposition of some $(t+1)$-concrete tree decomposition $\mathbf{T}'$ in $\mathcal{L}(\mathcal{A}(\varphi, t+1))$. By Theorem 4, $\mathbf{T}'$ is a $(t+1)$-concrete tree decomposition of some graph G' of treewidth at most t such that $G' \models \varphi$. Since $\mathbf{T}$ is a sub-decomposition of $\mathbf{T}'$, by Theorem 6, G is a subgraph of G'. Therefore, G' is a (φ, t)-supergraph of G. ◀

We note that Lemma 13 alone does not provide us with an efficient algorithm to determine whether a graph G has a (φ, t)-supergraph. If G does not admit such a supergraph, then no $(t+1)$-concrete tree decomposition G belongs to $\mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1)))$. However, if G does admit a (φ, t)-supergraph, then Theorem 6 only guarantees that some $(t+1)$-concrete tree decomposition $\mathbf{T}$ of G belongs to $\mathrm{Sub}(\mathcal{A}(\varphi, t+1))$. The problem is that G may have exponentially[3] many such decompositions, and we do not know a priori which of these should be tested for membership in $\mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1)))$.

The issue described above can be remedied with the results from Section 5. More specifically, from Theorem 12 we have that for any given connected graph G of treewidth t and maximum degree Δ, one can construct a tree automaton $\mathcal{A}(G, t+1)$ over $\mathcal{B}(t+1)$ which accepts a $(t+1)$-concrete tree decomposition $\mathbf{T}$ if and only if the graph $\mathcal{G}(\mathbf{T})$ is isomorphic to G. Therefore, a connected graph G has a (φ, t)-supergraph if and only if

$$\mathcal{L}(\mathcal{A}(G, t+1)) \cap \mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1))) \neq \emptyset. \tag{4}$$

The next theorem states that Equation 4 yields an efficient algorithm for testing whether connected graphs of bounded degree have a (φ, t)-supergraph.

▶ **Theorem 14** (Main Theorem)**.** *There is a computable function f, and an algorithm $\mathfrak{A}$ that takes as input a CMSO sentence φ, a positive integer t, and a connected graph G of maximum degree Δ, and determines in time $f(|\varphi|, t) \cdot 2^{O(\Delta \cdot t)} \cdot |G|^{O(t)}$ whether G has a (φ, t)-supergraph.*

Proof. By Lemma 13, G has a (φ, t)-supergraph if and only if there exists some $\mathbf{T} \in \mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1)))$ such that $\mathbf{T}$ is a $(t+1)$-concrete tree decomposition of G. By Theorem 12, $\mathcal{L}(\mathcal{A}(G, t+1))$ accepts a $(t+1)$-tree decomposition of G if and only if $\mathcal{G}(\mathbf{T})$ is isomorphic to G. Therefore, G has a (φ, t)-supergraph if and only the intersection of $\mathcal{L}(\mathcal{A}(G, t+1))$ with $\mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1)))$ is nonempty.

By Theorem 12, the tree-automaton $\mathcal{A}(G, t+1)$ can be constructed in time $2^{O(\Delta \cdot t)} \cdot |G|^{O(t)}$, and therefore the size of $\mathcal{A}(G, t+1)$ is bounded by $2^{O(\Delta \cdot t)} \cdot |G|^{O(t)}$. By Theorem 6 and Theorem 4, the tree-automaton $\mathrm{Sub}(\mathcal{A}(\varphi, t+1))$ can be constructed in time $f(|\varphi|, t)$ for some computable function $f : \mathbb{N}^2 \to \mathbb{N}$, and therefore, the size of $\mathrm{Sub}(\mathcal{A}(\varphi, t))$ is bounded by $f(|\varphi|, t)$.

[3] In fact a graph G of treewidth t may have infinitely many $(t+1)$-concrete tree decompositions, but we only need to consider those which have at most $|V_G| + |E_G|$ nodes.

Finally, given tree automata $\mathcal{A}_1$ and $\mathcal{A}_2$, one can determine whether $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) \neq \emptyset$ in time $O(|\mathcal{A}_1| \cdot |\mathcal{A}_2|)$ (Lemma 1). In particular, one can determine whether $\mathcal{L}(\mathcal{A}(G, t+1)) \cap \mathcal{L}(\mathrm{Sub}(\mathcal{A}(\varphi, t+1))) \neq \emptyset$ in time $f(|\varphi|, t) \cdot 2^{O(\Delta \cdot t)} \cdot |G|^{O(t)}$. □ ◀

7 Contraction Closed Graph Parameters

In this section we deal with simple graphs, i.e., graphs without loops or multiple edges. Therefore, we may write $\{u, v\}$ to denote an edge e whose endpoints are u and v.

Let G be a graph and $\{u, v\}$ be an edge of G. We let G/uv denote the graph that is obtained from G by deleting the edge $\{u, v\}$ and by merging vertices u and v into a single vertex x_{uv}. We say that G/uv is obtained from G by an edge-contraction. We say that a graph G' is a *contraction* of G if G' is obtained from G by a sequence of edge contractions. We say that G' is a minor of G if G' is a contraction of some subgraph of G. We say that a graph G is an appex graph if after deleting some vertex of G the resulting graph is planar.

A graph parameter is a function $\mathbf{p}$ mapping graphs to non-negative integers in such a way that $\mathbf{p}(G) = \mathbf{p}(G')$ whenever G is isomorphic to G'. We say that $\mathbf{p}$ is contraction closed if $\mathbf{p}(G') \leq \mathbf{p}(G)$ whenever G' is a contraction of G.

A graph property is simply a set $\mathcal{P}$ of graphs. We say that a property $\mathcal{P}$ is contraction-closed if for every two graphs G, G' for which G' is a contraction of G, the fact that $G \in \mathcal{P}$ implies that $G' \in \mathcal{P}$.

7.1 Diameter Improvement Problems

Let u and v be vertices in an graph G. The distance from u to v, denoted by $dist(u, v)$ is the number of edges in the shortest path from u to v. If no such path exists, we set $dist(u, v) = \infty$. The diameter of G is defined as $diam(G) = \max_{u,v} dist(u, v)$. In the PLANAR DIAMETER IMPROVEMENT problem (PDI), we are given an graph G and a positive integer d, and the goal is to determine whether G has a planar supergraph G' of diameter at most d. As mentioned in the introduction, there is an algorithm that solves the PDI problem in time $f(d) \cdot |G|^{O(1)}$, where $f : \mathbb{N} \to \mathbb{N}$ is not known to be computable. Additionally, even the problem of determining whether PDI admits an algorithm running in time $f_1(d) \cdot |G|^{f_2(d)}$ for computable functions f_1, f_2 remains open for more than two decades [13, 5]. The next theorem solves this problem in when the input graphs are connected and have bounded degree.

▶ **Theorem 15.** *There is a computable function $f : \mathbb{N} \to \mathbb{N}$, and an algorithm $\mathfrak{A}$ that takes as input, a positive integer d, and a connected graph G of maximum degree Δ, and determines in time $f(d) \cdot 2^{O(\Delta \cdot d)} \cdot |G|^{O(d)}$ whether G has a planar supergraph G' of diameter at most d.*

Proof. It should be clear that there is an algorithm that takes a positive integer d as input, and constructs, in time $O(d)$, a CMSO formula $Diam_d$ which is true on a graph G' if and only if G' has diameter at most d. Additionally, using Kuratowski's theorem, and the fact that minor relation is CMSO expressible, one can define a CMSO formula *Planar* which is true on a graph G' if and only if G' is planar. Finally, it can be shown that any planar graph of diameter at most d has treewidth $O(d)$. Therefore, by setting $\varphi = Diam_d \wedge Planar$, $t = O(d)$, and by renaming $f(|\varphi|, t)$ to $f(d)$ in Theorem 14, we have an algorithm running in time $f(d) \cdot 2^{O(\Delta \cdot d)} \cdot |G|^{O(d)}$ to determine whether G has a planar supergraph G' of diameter at most d. ◀

We note that the algorithm $\mathfrak{A}$ of Theorem 15 does not impose any restriction on the degree of a prospective supergraph G' of G. Theorem 15 can be generalized to the setting of graphs of constant genus as follows.

▶ **Theorem 16.** *There is a computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and an algorithm $\mathfrak{A}$ that takes as input, positive integers d, g, and a connected graph G of maximum degree Δ, and determines in time $f(d,g) \cdot 2^{O(\Delta \cdot d)} \cdot |G|^{O(d \cdot g)}$ whether G has a supergraph G' of genus at most g and diameter at most d.*

A graph is 1-outerplanar if it can be embedded in the plane in such a way that every vertex lies in the outer face of the embedding. A graph is k-outerplanar if it can be embedded in the plane in such a way that after deleting all vertices in the outer face, the remaining graph is $(k-1)$-outerplanar. In [5] Cohen et al. have considered the k-OUTERPLANAR DIAMETER IMPROVEMENT problem (k-OPDI), a variant of the PDI problem in which the target supergraph is required to be k-outerplanar instead of planar. In particular, they have shown that the 1-OPDI problem can be solved in polynomial time. The complexity of the k-OPDI problem with respect to explicit algorithms was left as an open problem for $k \geq 2$. The next theorem states that for each fixed k, k-OPDI is strongly uniformly fixed parameter tractable with respect to the parameter d on connected graphs of bounded degree.

▶ **Theorem 17.** *There is a computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and an algorithm $\mathfrak{A}$ that takes as input, positive integers d, k, and a connected graph G of maximum degree Δ, and determines in time $f(k,d) \cdot 2^{O(\Delta \cdot k)} \cdot |G|^{O(k)}$ whether G has a k-outerplanar supergraph G' of diameter at most d.*

Finally, the SERIES-PARALLEL DIAMETER IMPROVEMENT problem (SPDI) consists in determining whether a graph G has a series parallel supergraph of diameter at most d. The parameterized complexity of this problem was left as an open problem in [5]. The next theorem states that SPDI is strongly uniformly fixed parameter tractable with respect to the parameter d on connected graphs of bounded degree.

▶ **Theorem 18.** *There is a computable function $f : \mathbb{N} \to \mathbb{N}$, and an algorithm $\mathfrak{A}$ that takes as input, a positive integer d and a connected graph G of maximum degree Δ, and determines in time $f(d) \cdot 2^{O(\Delta)} \cdot |G|^{O(1)}$ whether G has a series-parallel supergraph G' of diameter at most d.*

7.2 Contraction Bidimensional Parameters

Fomin, Golovach and Thilikos [15] have defined a sequence $\{\leqq_k\}_{k \in \mathbb{N}}$ of graphs and have shown that these graphs serve as obstructions for small treewidth on H-minor free graphs, whenever H is an appex graph. More precisely, they have proved the following result.

▶ **Theorem 19** (Fomin-Golovach-Thilikos [15])**.** *For every apex graph H, there is a $c_H > 0$ such that every connected H-minor-free graph of treewidth at least $c_H \cdot k$ contains $\leqq_k$ as a contraction.*

We say that a graph parameter $\mathbf{p}$ is Gamma-unbounded if there is a computable function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\alpha \in \omega(1)$, and $\mathbf{p}(\leqq_k) \geq \alpha(k)$ for every $k \in \mathbb{N}$.

We say that a parameter $\mathbf{p}$ is effectively CMSO definable if there is a computable function $f : \mathbb{N} \to \mathbb{N}$, and an algorithm $\mathfrak{A}$ that takes as input a positive integer k and constructs, in time at most $f(k)$, a CMSO-sentence φ which is true on an graph G if and only if $\mathbf{p}(G) \leq k$. The following theorem is a corollary of Theorem 14 and Theorem 19.

▶ **Theorem 20.** *Let $\mathbf{p}$ be a Gamma-unbounded effectively CMSO definable graph parameter, and let $\mathcal{P}$ be a CMSO definable graph property excluding some appex graph H as a minor.*

Then there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm $\mathfrak{A}$ that takes as input a positive integer k, and a connected graph G of maximum degree Δ, and determines, in time $f(k) \cdot 2^{O(\Delta \cdot f(k))} \cdot |G|^{f(k)}$, whether G has a supergraph G' such that $G' \in \mathcal{P}$ and $\mathbf{p}(G') \leq k$.

Note that similarly to the case of diameter improvement problem, if $\mathbf{p}$ is an unbounded effectively CMSO definable graph parameter, then we can determine whether a graph G has an r-outerplanar supergraph G' with $\mathbf{p}(G') \leq k$ in time $f(r,k) \cdot 2^{O(\Delta \cdot r)} \cdot |G|^{O(r)}$ for some computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. In other words, this problem, for connected bounded degree graphs, is strongly uniformly fixed parameter tractable with respect to the parameter $\mathbf{p}$ for each fixed r.

▶ **Definition 21.** A graph parameter $\mathbf{p}$ is contraction-bidimensional if the following conditions are satisfied.

1. $\mathbf{p}$ is contraction-closed.
2. If G is a graph which has $\leqslant_{\pm k}$ as a contraction, then $\mathbf{p}(G) \geq \Omega(k^2)$.

For instance, the following parameters are contraction bidimensional.

1. Size of a vertex cover.
2. Size of a feedback vertex set.
3. Size of a minimum maximal matching.
4. Size of a dominating set.
5. Size of a edge dominating set.
6. Size of a clique traversal set.

▶ **Theorem 22** ([15, 16]). *Let $\mathbf{p}$ be a bidimensional parameter. Then if $\mathbf{p}(G) \leq k$, the treewidth of $\mathbf{p}$ is at most $O(\sqrt{k})$.*

▶ **Theorem 23.** *For each effectively CMSO-definable contraction-bidimensional parameter $\mathbf{p}$, there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm $\mathfrak{A}$ that takes as input a positive integer k, and a connected graph G of maximum degree Δ, and determines in time $f(k) \cdot 2^{O(\Delta \cdot \sqrt{k})} \cdot |G|^{O(\sqrt{k})}$ whether G has a planar supergraph G' with $\mathbf{p}(G') \leq k$.*

For instance, Theorem 23 states that for some computable function $f : \mathbb{N} \to \mathbb{N}$, one can determine in time $f(k) \cdot 2^{O(\Delta \cdot \sqrt{k})} \cdot |G|^{O(\sqrt{k})}$ whether G has a planar supergraph G' with feedback vertex set at most k. We note that in view of Theorem 22, the planarity requirement of Theorem 23 can be replaced for any CMSO definable property $\mathcal{P}$ which excludes some apex graph as a minor.

Acknowledgements. This work was supported by the Bergen Research Foundation. The author thanks Michael Fellows, Fedor Fomin, Petr Golovach, Daniel Lokshtanov and Saket Saurabh for interesting discussions.

References

1 Karl Abrahamson and Michael Fellows. Finite automata, bounded treewidth, and well-quasiordering. *Contemporary Mathematics*, 147:539–539, 1993.
2 Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proc. of SODA 2008*, pages 641–650. SIAM, 2008.
3 Mikołaj Bojańczyk and Michal Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In *Proc. of LICS 2016*, pages 407–416. ACM, 2016.
4 Mikołaj Bojańczyk and Michal Pilipczuk. Optimizing tree decompositions in MSO. In *Proc. of STACS 2017 (To appear)*, 2017.

5 Nathann Cohen, Daniel Gonçalves, Eunjung Kim, Christophe Paul, Ignasi Sau, Dimitrios M. Thilikos, and Mathias Weller. A polynomial-time algorithm for outerplanar diameter improvement. In *Proc. of the 10th International Computer Science Symposium in Russia (CSR 2015)*, volume 9139 of *LNCS*, pages 123–142, 2015.

6 H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available at `http://www.grappa.univ-lille3.fr/tata`, 2007. Release October, 12th 2007.

7 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

8 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic. A language-theoretic approach.* HAL, June 14 2012. URL: `http://hal.archives-ouvertes.fr/hal-00646514`.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

10 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

11 Michael Elberfeld. Context-free graph properties via definable decompositions. In *Proc. of the 25th Conference on Computer Science Logic (CSL 2016)*, volume 62 of *LIPIcs*, pages 17:1–17:16, 2016.

12 Michael R. Fellows and Rodney G. Downey. Parameterized computational feasibility. *Feasible Mathematics II*, 13:219–244, 1995.

13 Michael R. Fellows and Michael A. Langston. On search decision and the efficiency of polynomial-time algorithms. In *Proc. of STOC 1989*, pages 501–512. ACM, 1989.

14 Jörg Flum, Markus Frick, and Martin Grohe. Query evaluation via tree-decompositions. *Journal of the ACM (JACM)*, 49(6):716–752, 2002.

15 Fedor V. Fomin, Petr Golovach, and Dimitrios M. Thilikos. Contraction obstructions for treewidth. *Journal of Combinatorial Theory, Series B*, 101(5):302–314, 2011.

16 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *Proc. of SODA 2010*, pages 503–510, 2010.

17 Neil Robertson and Paul D. Seymour. Graph minors. XIII. the disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.

18 Neil Robertson and Paul D Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

Inductive and Functional Types in Ludics

Alice Pavaux

Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030, Paris, France

Abstract

Ludics is a logical framework in which types/formulas are modelled by sets of terms with the same computational behaviour. This paper investigates the representation of inductive data types and functional types in ludics. We study their structure following a game semantics approach. Inductive types are interpreted as least fixed points, and we prove an internal completeness result giving an explicit construction for such fixed points. The interactive properties of the ludics interpretation of inductive and functional types are then studied. In particular, we identify which higher-order functions types fail to satisfy type safety, and we give a computational explanation.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic

Keywords and phrases Ludics, Inductive types, Fixed point, Linear logic, Game semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.34

1 Introduction

1.1 Context and Contributions

Context. *Ludics* was introduced by Girard [10] as a variant of *game semantics* with interactive types. Game Semantics has successfully provided fully abstract models for various logical systems and programming languages, among which PCF [11]. Although very close to Hyland–Ong (HO) games, ludics reverses the approach: in HO games one defines first the interpretation of a type (an arena) before giving the interpretation for the terms of that type (the strategies), while in ludics the interpretation of terms (the *designs*) is primitive and the types (the *behaviours*) are recovered dynamically as well-behaved sets of terms. This approach to types is similar to what exists in realisability [12] or geometry of interaction [9].

The motivation for such a framework was to reconstruct logic around the dynamics of proofs. Girard provides a ludics model for (a polarised version of) multiplicative-additive linear logic (MALL); a key role in his interpretation of logical connectives is played by the *internal completeness* results, which allow for a direct description of the behaviours' content. As most behaviours are not the interpretation of MALL formulas, an interesting question, raised from the beginning of ludics, is whether these remaining behaviours can give a logical counterpart to computational phenomena. In particular, data and functions [16, 15], and also fixed points [2] have been studied in the setting of ludics. The present work follows this line of research.

Real life (functional) programs usually deal with data, functions over it, functions over functions, etc. *Data types* allow one to present information in a structured way. Some data types are defined *inductively*, for example:

Listing 1 Example of inductive types in OCaml

```
> type nat = Zero | Succ of nat ;;
> type 'a list = Nil | Cons of 'a * 'a list ;;
> type 'a tree = Empty | Node of 'a * ('a tree) list ;;
```

Upon this basis we can consider *functional types*, which are either first-order – from data to data – or higher-order – i.e., taking functions as arguments or returning functions as a result. This article aims at interpreting constructively the (potentially inductive) data types and the (potentially higher-order) functional types as behaviours of ludics, so as to study their structural properties. Inductive types are defined as (least) *fixed points*. As pointed out by Baelde, Doumane and Saurin [2], the fact that ludics puts the most constraints on the formation of terms instead of types, conversely to game semantics, makes it a more natural setting for the interpretation of fixed points than HO games [4].

Contributions. The main contributions of this article are the following:

- We prove that internal completeness holds for infinite unions of behaviours satisfying particular conditions (Theorem 30), leading to an explicit construction of the least fixed points in ludics (Proposition 34).
- Inductive and functional types are interpreted as behaviours, and we prove that such behaviours are *regular* (Corollary 35 and Proposition 42). Regularity (that we discuss more in § 1.2) is a property that could be used to characterise the behaviours corresponding to μMALL formulas [1, 2] – i.e., MALL with fixed points.
- We show that a functional behaviour fails to satisfy *purity*, a property ensuring the safety of all possible executions (further explained in § 1.2), if and only if it is higher order and takes functions as argument (Proposition 43); this is typically the case of $(\mathbf{A} \multimap \mathbf{B}) \multimap \mathbf{C}$. In § 5.2 we discuss the computational meaning of this result.

The present work is conducted in the term-calculus reformulation of ludics by Terui [16] restricted to the linear part – the idea is that programs call each argument at most once.

Related Work. The starting point for our study of inductive types as fixed points in ludics is the work by Baelde, Doumane and Saurin [2]. In their article, they provide a ludics model for μMALL, a variant of multiplicative-additive linear logic with least and greatest fixed points. The existence of fixed points in ludics is ensured by Knaster-Tarski theorem, but this approach does not provide an explicit way to construct the fixed points; we will consider Kleene fixed point theorem instead. Let us also mention the work of Melliès and Vouillon [13] which introduces a realisability model for recursive (i.e., inductive and coinductive) polymorphic types.

The representation of both data and functions in ludics has been studied previously. Terui [16] proposes to encode them as designs in order to express computability properties in ludics, but data and functions are not considered at the level of behaviours. Sironi [15] describes the behaviours corresponding to some data types: integers, lists, records, etc. as well as first-order function types; our approach generalises hers by considering generic data types and also higher order functions types.

1.2 Background

Behaviours and Internal Completeness. A behaviour $\mathbf{B}$ is a set of designs which pass the same set of tests $\mathbf{B}^\perp$, where tests are also designs. $\mathbf{B}^\perp$ is called the *orthogonal* of $\mathbf{B}$, and behaviours are closed under bi-orthogonal: $\mathbf{B}^{\perp\perp} = \mathbf{B}$. New behaviours can be formed upon others using various constructors. In this process, internal completeness, which can be seen as a built-in notion of observational equivalence, ensures that two agents reacting the same way to any test are actually equal. From a technical point of view, this means that it is not necessary to apply a $\perp\perp$-closure for the sets constructed to be behaviours.

Paths: Ludics as Game Semantics. This paper makes the most of the resemblance between ludics and HO game semantics. The connections between them have been investigated in many pieces of work [3, 6, 7] where designs are described as (innocent) strategies, i.e., in terms of the traces of their possible interactions. Following this idea, Fouqueré and Quatrini define *paths* [7], corresponding to legal plays in HO games, and they characterise a behaviour by its set of *visitable paths*. This is the approach we follow. The definitions of regularity and purity rely on paths, since they are properties of the possible interactions of a behaviour.

Regularity: Towards a Characterisation of μMALL? Our proof that internal completeness holds for an infinite union of increasingly large behaviours (Theorem 30) relies in particular on the additional hypothesis of regularity for these behaviours. Intuitively, a behaviour $\mathbf{B}$ is regular if every path in a design of $\mathbf{B}$ is realised by interacting with a design of $\mathbf{B}^{\perp}$, and vice versa. This property is not actually ad hoc: it was introduced by Fouqueré and Quatrini [8] to characterise the denotations of MALL formulas as being precisely the regular behaviours satisfying an additional finiteness condition. In this direction, our intuition is that – forgetting about finiteness – regularity captures the behaviours corresponding to formulas of μMALL. Although such a characterisation is not yet achieved, we provide a first step by showing that the *data patterns*, a subset of positive μMALL formulas, yield only regular behaviours (Proposition 33).

Purity: Type Safety. Ludics has a special feature for termination which is not present in game semantics: the *daimon* ✠. On a computational point of view, the daimon is commonly interpreted as an error, an exception raised at run-time causing the program to stop (see for example the notes of Curien [5]). Thinking of Ludics as a programming language, we would like to guarantee *type safety*, that is, ensure that "well typed programs cannot go wrong" [14]. This is the purpose of purity, a property of behaviours: in a pure behaviour, maximal interaction traces are ✠-free, in other words whenever the interaction stops with ✠ it is actually possible to "ask for more" and continue the computation. Introduced by Sironi [15] (and called *principality* in her work), this property is related to the notions of *winning* designs [10] and *pure* designs [16], but at the level of a behaviour. As expected, data types are pure (Corollary 40), but not always functional types are; we identify the precise cases where impurity arises (Proposition 43), and explain why some types are not safe.

1.3 Outline

In Section 2 we present ludics and we state internal completeness for the logical connectives constructions. In Section 3 we recall the notion of path, so as to define formally regularity and purity and prove their stability under the connectives. Section 4 studies inductive data types, which we interpret as behaviours; Kleene theorem and internal completeness for infinite union allows us to give an explicit and direct construction for the least fixed point, with no need for bi-orthogonal closure; we deduce that data types are regular and pure. Finally, in Section 5, we study functional types, showing in what case purity fails.

2 Computational Ludics

This section introduces the ludics background necessary for the rest of the paper, in the formalism of Terui [16]. The *designs* are the primary objects of ludics, corresponding to (polarised) proofs or programs in a Curry-Howard perspective. Cuts between designs can occur, and their reduction is called *interaction*. The *behaviours*, corresponding to the types

or formulas of ludics, are then defined thanks to interaction. Compound behaviours can be formed with *logical connectives* constructions which satisfy *internal completeness*.

2.1 Designs and Interaction

Suppose given a set of variables $\mathcal{V}_0$ and a set $\mathcal{S}$, called **signature**, equipped with an arity function $ar : \mathcal{S} \to \mathbb{N}$. Elements $a, b, \dots \in \mathcal{S}$ are called **names**. A **positive action** is either ✠ (daimon), Ω (divergence), or $\overline{a}$ with $a \in \mathcal{S}$; a **negative action** is $a(x_1, \dots, x_n)$ where $a \in \mathcal{S}$, $\mathrm{ar}(a) = n$ and $x_1, \dots, x_n \in \mathcal{V}_0$ distinct. An action is **proper** if it is neither ✠ nor Ω.

▶ **Definition 1.** Positive and negative **designs**[1] are coinductively defined by:

$$\mathfrak{p} ::= ✠ \mid \Omega \mid x|\overline{a}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_{\mathrm{ar}(a)} \rangle \mid \mathfrak{n}_0|\overline{a}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_{\mathrm{ar}(a)} \rangle$$
$$\mathfrak{n} ::= \sum_{a \in \mathcal{S}} a(x^a_1, \dots, x^a_{\mathrm{ar}(a)}).\mathfrak{p}_a$$

Positive designs play the same role as *applications* in λ-calculus, and negative designs the role of *abstractions*, where each name $a \in \mathcal{S}$ binds $\mathrm{ar}(a)$ variables.

Designs are considered up to α-equivalence. We will often write $a(\overrightarrow{x})$ (resp. $\overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle$) instead of $a(x_1, \dots, x_n)$ (resp. $\overline{a}\langle \mathfrak{n}_1 \dots \mathfrak{n}_n \rangle$). Negative designs can be written as partial sums, for example $a(x, y).\mathfrak{p} + b().\mathfrak{q}$ instead of $a(x, y).\mathfrak{p} + b().\mathfrak{q} + \sum_{c \neq a, c \neq b} c(\overrightarrow{z^c}).\Omega$.

Given a design $\mathfrak{d}$, the definitions of the **free variables** of $\mathfrak{d}$, written $\mathrm{fv}(\mathfrak{d})$, and the (capture-free) **substitution** of x by a negative design $\mathfrak{n}$ in $\mathfrak{d}$, written $\mathfrak{d}[\mathfrak{n}/x]$, can easily be inferred. The design $\mathfrak{d}$ is **closed** if it is positive and it has no free variable. A **subdesign** of $\mathfrak{d}$ is a subterm of $\mathfrak{d}$. A **cut** in $\mathfrak{d}$ is a subdesign of $\mathfrak{d}$ of the form $\mathfrak{n}_0|\overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle$, and a design is **cut-free** if it has no cut.

In the following, we distinguish a particular variable x_0, that cannot be bound. A positive design $\mathfrak{p}$ is **atomic** if $\mathrm{fv}(\mathfrak{p}) \subseteq \{x_0\}$; a negative design $\mathfrak{n}$ is **atomic** if $\mathrm{fv}(\mathfrak{n}) = \emptyset$.

A design is **linear** if for every subdesign of the form $x|\overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle$ (resp. $\mathfrak{n}_0|\overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle$), the sets $\{x\}$, $\mathrm{fv}(\mathfrak{n}_1)$, …, $\mathrm{fv}(\mathfrak{n}_{\mathrm{ar}(a)})$ (resp. the sets $\mathrm{fv}(\mathfrak{n}_0)$, $\mathrm{fv}(\mathfrak{n}_1)$, …, $\mathrm{fv}(\mathfrak{n}_{\mathrm{ar}(a)})$) are pairwise disjoint. This article focuses on linearity, so in the following when writing "design" we mean "linear design".

▶ **Definition 2.** The **interaction** corresponds to reduction steps applied on cuts:

$$\sum_{a \in \mathcal{S}} a(x^a_1, \dots, x^a_{\mathrm{ar}(a)}).\mathfrak{p}_a \mid \overline{b}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_k \rangle \quad \rightsquigarrow \quad \mathfrak{p}_b[\mathfrak{n}_1/x^b_1, \dots, \mathfrak{n}_k/x^b_k]$$

We will later describe an interaction as a sequence of actions, a path (Definition 13).

Let $\mathfrak{p}$ be a design, and let $\rightsquigarrow^*$ denote the reflexive transitive closure of $\rightsquigarrow$; if there exists a design $\mathfrak{q}$ which is neither a cut nor Ω and such that $\mathfrak{p} \rightsquigarrow^* \mathfrak{q}$, we write $\mathfrak{p} \Downarrow \mathfrak{q}$; otherwise we write $\mathfrak{p} \Uparrow$. The normal form of a design, defined below, exists and is unique [16].

▶ **Definition 3.** The **normal form** of a design $\mathfrak{d}$, noted $(\!|\mathfrak{d}|\!)$, is defined by:

$$(\!|\mathfrak{p}|\!) = ✠ \quad \text{if } \mathfrak{p} \Downarrow ✠ \qquad (\!|\mathfrak{p}|\!) = x|\overline{a}\langle (\!|\mathfrak{n}_1|\!), \dots, (\!|\mathfrak{n}_n|\!) \rangle \quad \text{if } \mathfrak{p} \Downarrow x|\overline{a}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_n \rangle$$
$$(\!|\mathfrak{p}|\!) = \Omega \quad \text{if } \mathfrak{p} \Uparrow \qquad (\!|\sum_{a \in \mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a|\!) = \sum_{a \in \mathcal{S}} a(\overrightarrow{x^a}).(\!|\mathfrak{p}_a|\!)$$

Note that the normal form of a closed design is either ✠ (convergence) or Ω (divergence). Orthogonality expresses the convergence of the interaction between two atomic designs, and behaviours are sets of designs closed by bi-orthogonal.

[1] In the following, the symbols $\mathfrak{d}, \mathfrak{e}, \dots$ refer to designs of any polarity, while $\mathfrak{p}, \mathfrak{q}, \dots$ and $\mathfrak{m}, \mathfrak{n}, \dots$ are specifically for positive and negative designs respectively.

▶ **Definition 4.** Two atomic designs $\mathfrak{p}$ and $\mathfrak{n}$ are **orthogonal**, noted $\mathfrak{p} \perp \mathfrak{n}$, if $(\!|\mathfrak{p}[\mathfrak{n}/x_0]|\!) = \maltese$.

Given an atomic design $\mathfrak{d}$, define $\mathfrak{d}^\perp = \{\mathfrak{e} \mid \mathfrak{d} \perp \mathfrak{e}\}$; if E is a set of atomic designs of same polarity, define $E^\perp = \{\mathfrak{d} \mid \forall \mathfrak{e} \in E, \mathfrak{d} \perp \mathfrak{e}\}$.

▶ **Definition 5.** A set $\mathbf{B}$ of atomic designs of same polarity is a **behaviour**[2] if $\mathbf{B}^{\perp\perp} = \mathbf{B}$. A behaviour is either positive or negative depending on the polarity of its designs.

Behaviours could alternatively be defined as the orthogonal of a set E of atomic designs of same polarity – E corresponds to a set of *tests* or *trials*. Indeed, $E^\perp$ is always a behaviour, and every behaviour $\mathbf{B}$ is of this form by taking $E = \mathbf{B}^\perp$.

The *incarnation* of a behaviour $\mathbf{B}$ contains the cut-free designs of $\mathbf{B}$ whose actions are all visited during an interaction with a design in $\mathbf{B}^\perp$. Those correspond to the cut-free designs that are minimal for the **stable ordering** $\sqsubseteq$, where $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ if $\mathfrak{d}$ can be obtained from $\mathfrak{d}'$ by substituting positive subdesigns for some occurrences of Ω.

▶ **Definition 6.** Let $\mathbf{B}$ be a behaviour and $\mathfrak{d} \in \mathbf{B}$ cut-free.

- The **incarnation** of $\mathfrak{d}$ in $\mathbf{B}$, written $|\mathfrak{d}|_{\mathbf{B}}$, is the smallest (for $\sqsubseteq$) cut-free design $\mathfrak{d}'$ such that $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ and $\mathfrak{d}' \in \mathbf{B}$. If $|\mathfrak{d}|_{\mathbf{B}} = \mathfrak{d}$ we say that $\mathfrak{d}$ is **incarnated** in $\mathbf{B}$.
- The **incarnation** $|\mathbf{B}|$ of $\mathbf{B}$ is the set of the (cut-free) incarnated designs of $\mathbf{B}$.

2.2 Logical Connectives

Behaviour constructors – the *logical connectives* – can be applied so as to form compound behaviours. These connectives, coming from (polarised) linear logic, are used for interpreting formulas as behaviours, and will also indeed play the role of type constructors for the types of data and functions. In this subsection, after defining the connectives we consider, we state the *internal completeness* theorem for these connectives.

Let us introduce some notations. In the rest of this article, suppose the signature $\mathcal{S}$ contains distinct unary names $\blacktriangle, \pi_1, \pi_2$ and a binary name $\wp$, and write $\blacktriangledown = \overline{\blacktriangle}, \iota_1 = \overline{\pi}_1, \iota_2 = \overline{\pi}_2$ and $\bullet = \overline{\wp}$. Given a behaviour $\mathbf{B}$ and x fresh, define $\mathbf{B}^x = \{\mathfrak{d}[x/x_0] \mid \mathfrak{d} \in \mathbf{B}\}$; such a substitution operates a "delocation" with no repercussion on the behaviour's inherent properties. Given a k-ary name $a \in \mathcal{S}$, we write $\overline{a}\langle \mathbf{N}_1, \dots, \mathbf{N}_k \rangle$ or even $\overline{a}\langle \overrightarrow{\mathbf{N}} \rangle$ for $\{x_0 | \overline{a}\langle \overrightarrow{\mathfrak{n}} \rangle \mid \mathfrak{n}_i \in \mathbf{N}_i\}$, and write $a(\overrightarrow{x}).\mathbf{P}$ for $\{a(\overrightarrow{x}).\mathfrak{p} \mid \mathfrak{p} \in \mathbf{P}\}$. For a negative design $\mathfrak{n} = \sum_{a \in \mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ and a name $a \in \mathcal{S}$, we denote by $\mathfrak{n}\restriction a$ the design $a(\overrightarrow{x^a}).\mathfrak{p}_a$ (that is $a(\overrightarrow{x^a}).\mathfrak{p}_a + \sum_{b \neq a} b(\overrightarrow{x^b}).\Omega$).

▶ **Definition 7** (Logical connectives)**.**

$\downarrow\mathbf{N} = \blacktriangledown\langle \mathbf{N} \rangle^{\perp\perp}$	(**positive shift**)
$\uparrow\mathbf{P} = (\blacktriangle(x).\mathbf{P}^x)^{\perp\perp}$, with x fresh	(**negative shift**)
$\mathbf{M} \oplus \mathbf{N} = (\iota_1\langle \mathbf{M} \rangle \cup \iota_2\langle \mathbf{N} \rangle)^{\perp\perp}$	(**plus**)
$\mathbf{M} \otimes \mathbf{N} = \bullet\langle \mathbf{M}, \mathbf{N} \rangle^{\perp\perp}$	(**tensor**)
$\mathbf{N} \multimap \mathbf{P} = (\mathbf{N} \otimes \mathbf{P}^\perp)^\perp$	(**linear map**)

Our connectives $\downarrow$, $\uparrow$, $\oplus$ and $\otimes$ match exactly those defined by Terui [16], who also proves the following internal completeness theorem stating that connectives apply on behaviours in a constructive way – there is no need to close by bi-orthogonal. For each connective, we

[2] Symbols $\mathbf{A}, \mathbf{B}, \dots$ will designate behaviours of any polarity, while $\mathbf{M}, \mathbf{N} \dots$ and $\mathbf{P}, \mathbf{Q}, \dots$ will be for negative and positive behaviours respectively.

present two versions of internal completeness: one concerned with the full behaviour, the other with the behaviour's incarnation.

▶ **Theorem 8** (Internal completeness for connectives)**.**

$$\downarrow\mathbf{N} = \blacktriangledown\langle\mathbf{N}\rangle \cup \{\maltese\} \qquad |\downarrow\mathbf{N}| = \blacktriangledown\langle|\mathbf{N}|\rangle \cup \{\maltese\}$$
$$\uparrow\mathbf{P} = \{\mathfrak{n} \mid \mathfrak{n}|\blacktriangle \in \blacktriangle(x).\mathbf{P}^x\} \qquad |\uparrow\mathbf{P}| = \blacktriangle(x).|\mathbf{P}^x|$$
$$\mathbf{M} \oplus \mathbf{N} = \iota_1\langle\mathbf{M}\rangle \cup \iota_2\langle\mathbf{N}\rangle \cup \{\maltese\} \qquad |\mathbf{M} \oplus \mathbf{N}| = \iota_1\langle|\mathbf{M}|\rangle \cup \iota_2\langle|\mathbf{N}|\rangle \cup \{\maltese\}$$
$$\mathbf{M} \otimes \mathbf{N} = \bullet\langle\mathbf{M}, \mathbf{N}\rangle \cup \{\maltese\} \qquad |\mathbf{M} \otimes \mathbf{N}| = \bullet\langle|\mathbf{M}|, |\mathbf{N}|\rangle \cup \{\maltese\}$$

3 Paths and Interactive Properties of Behaviours

Paths are sequences of actions recording the trace of a possible interaction. For a behaviour $\mathbf{B}$, we can consider the set of its *visitable paths* by gathering all the paths corresponding to an interaction between a design of $\mathbf{B}$ and a design of $\mathbf{B}^\perp$. This notion is needed for defining *regularity* and *purity* and proving that those two properties of behaviours are stable under (some) connectives constructions.

3.1 Paths

This subsection adapts the definitions of path and visitable path from [7] to the setting of computational ludics. In order to do so, we need first to recover *location* in actions so as to consider sequences of actions.

Location is a primitive idea in Girard's ludics [10] in which the places of a design are identified with *loci* or *addresses*, but this concept is not visible in Terui's presentation of designs-as-terms. We overcome this by introducing actions with more information on location, which we call *located actions*, and which are necessary to:

- represent cut-free designs as trees – actually, forests – in a satisfactory way,
- define views and paths.

▶ **Definition 9.** A **located action**[3] κ is one of: $\maltese \mid x|\overline{a}\langle x_1, \ldots, x_{\mathrm{ar}(a)}\rangle \mid a_x(x_1, \ldots, x_{\mathrm{ar}(a)})$ where in the last two cases (**positive proper** and **negative proper** respectively), $a \in \mathcal{S}$ is the **name** of κ, the variables $x, x_1, \ldots, x_{\mathrm{ar}(a)}$ are distinct, x is the **address** of κ and $x_1, \ldots, x_{\mathrm{ar}(a)}$ are the **variables bound by** κ.

In the following, "action" will always refer to a located action. Similarly to notations for designs, $x|\overline{a}\langle\overrightarrow{x}\rangle$ stands for $x|\overline{a}\langle x_1, \ldots, x_n\rangle$ and $a_x(\overrightarrow{x})$ for $a_x(x_1, \ldots, x_n)$.

▶ **Example 10.** We show how cut-free designs can be represented as trees of located actions in this example. Let $a^2, b^2, c^1, d^0 \in \mathcal{S}$, where exponents stand for arities. The following design is represented by the tree of Fig. 1.

$$\mathfrak{d} = a(x_1, x_2).(x_2|\overline{b}\langle a(x_3, x_4).\maltese + c(y_1).(y_1|\overline{d}\langle\rangle), c(y_2).(x_1|\overline{d}\langle\rangle)\rangle)$$

Such a representation is in general a forest: a negative design $\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ gives as many trees as there is $a \in \mathcal{S}$ such that $\mathfrak{p}_a \neq \Omega$. The distinguished variable x_0 is given as address to every negative root of a tree, and fresh variables are picked as addresses for negative actions bound by positive ones. This way, negative actions from the same subdesign,

[3] Located actions will often be denoted by symbol κ, sometimes with its polarity: κ^+ or κ^-.

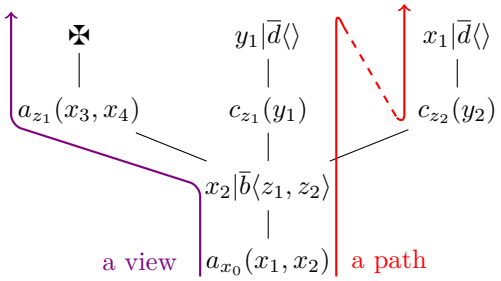


Figure 1 Representation of design $\mathfrak{d}$ from Example 10, with a path and a view of $\mathfrak{d}$.

i.e., part of the same sum, are given the same address. A tree is indeed to be read bottom-up: a proper action κ is **justified** if its address is bound by an action of opposite polarities appearing below κ in the tree; otherwise κ is called **initial**. Except the root of a tree, which is always initial, every negative action is justified by the only positive action immediately below it. If κ and κ' are proper, κ is **hereditarily justified** by κ' if there exist actions $\kappa_1, \ldots, \kappa_n$ such that $\kappa = \kappa_1$, $\kappa' = \kappa_n$ and for all i such that $1 \leq i < n$, κ_i is justified by κ_{i+1}.

Before giving the definitions of *view* and *path*, let us give an intuition. On Fig. 1 are represented a view and a path of design $\mathfrak{d}$. Views are branches in the tree representing a cut-free design (reading bottom-up), while paths are particular "promenades" starting from the root of the tree; not all such promenades are paths, though. Views correspond to *chronicles* in original ludics [10].

For every positive proper action $\kappa^+ = x|\overline{a}\langle\overrightarrow{y}\rangle$ define $\overline{\kappa^+} = a_x(\overrightarrow{y})$, and similarly if $\kappa^- = a_x(\overrightarrow{y})$ define $\overline{\kappa^-} = x|\overline{a}\langle\overrightarrow{y}\rangle$. Given a finite sequence of proper actions $s = \kappa_1 \ldots \kappa_n$, define $\overline{s} = \overline{\kappa_1} \ldots \overline{\kappa_n}$. Suppose now that if s contains an occurrence of ✠, it is necessarily in last position; the **dual** of s, written $\widetilde{s}$, is the sequence defined by:

- $\widetilde{s} = \overline{s}$✠ if s does not end with ✠,
- $\widetilde{s} = \overline{s'}$ if $s = s'$✠.

Note that $\widetilde{\widetilde{s}} = s$. The notions of **justified**, **hereditarily justified** and **initial** actions also apply in sequences of actions.

▶ **Definition 11.** An **alternated justified sequence** (or **aj-sequence**) s is a finite sequence of actions such that:

- (Alternation) Polarities of actions alternate.
- (Daimon) If ✠ appears, it is the last action of s.
- (Linearity) Each variable is the address of at most one action in s.

The (unique) justification of a justified action κ in an aj-sequence is noted $\mathrm{just}(\kappa)$, when there is no ambiguity on the sequence we consider.

▶ **Definition 12.** A **view** $\gtrapprox$ is an aj-sequence such that each negative action which is not the first action of $\gtrapprox$ is justified by the immediate previous action. Given a cut-free design $\mathfrak{d}$, $\gtrapprox$ is a **view of** $\mathfrak{d}$ if it is a branch in the representation of $\mathfrak{d}$ as a tree (modulo α-equivalence).

The way to extract **the view** of an aj-sequence is given inductively by:

- $\ulcorner\epsilon\urcorner = \epsilon$, where ϵ is the empty sequence,
- $\ulcorner s\kappa^+\urcorner = \ulcorner s\urcorner\kappa^+$,
- $\ulcorner s\kappa^-\urcorner = \ulcorner s_0\urcorner\kappa^-$ where s_0 is the prefix of s ending on $\mathrm{just}(\kappa^-)$, or $s_0 = \epsilon$ if κ^- initial.

The **anti-view** of an aj-sequence, noted $\llcorner s\lrcorner$, is defined symmetrically by reversing the role played by polarities; equivalently $\llcorner s\lrcorner = \widetilde{\ulcorner\widetilde{s}\urcorner}$.

▶ **Definition 13.** A **path** s is a positive-ended aj-sequence satisfying:
- (P-visibility) For all prefix $s'\kappa^+$ of s, $\text{just}(\kappa^+) \in \ulcorner s' \urcorner$
- (O-visibility) For all prefix $s'\kappa^-$ of s, $\text{just}(\kappa^-) \in \llcorner s' \lrcorner$

Given a cut-free design $\mathfrak{d}$, a path s is a **path of** $\mathfrak{d}$ if for all prefix s' of s, $\ulcorner s' \urcorner$ is a view of $\mathfrak{d}$.

Remark that the dual of a path is a path.

Paths are aimed at describing an interaction between designs. If $\mathfrak{d}$ and $\mathfrak{e}$ are cut-free atomic designs such that $\mathfrak{d} \perp \mathfrak{e}$, there exists a unique path s of $\mathfrak{d}$ such that $\tilde{s}$ is a path of $\mathfrak{e}$. We write this path $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$, and the good intuition is that it corresponds to the sequence of actions followed by the interaction between $\mathfrak{d}$ and $\mathfrak{e}$ on the side of $\mathfrak{d}$. An alternative way defining orthogonality is then given by the following proposition.

▶ **Proposition 14.** *$\mathfrak{d} \perp \mathfrak{e}$ if and only if there exists a path s of $\mathfrak{d}$ such that $\tilde{s}$ is a path of $\mathfrak{e}$.*

At the level a behaviour $\mathbf{B}$, the set of visitable paths describes all the possible interactions between a design of $\mathbf{B}$ and a design of $\mathbf{B}^\perp$.

▶ **Definition 15.** A path s is **visitable** in a behaviour $\mathbf{B}$ if there exist cut-free designs $\mathfrak{d} \in \mathbf{B}$ and $\mathfrak{e} \in \mathbf{B}^\perp$ such that $s = \langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle$. The set of visitable paths of $\mathbf{B}$ is written $V_{\mathbf{B}}$.

Note that for every behaviour $\mathbf{B}$, $\widetilde{V_{\mathbf{B}}} = V_{\mathbf{B}^\perp}$.

3.2 Regularity, Purity and Connectives

The meaning of regularity and purity has been discussed in the introduction. After giving the formal definitions, we prove that regularity is stable under all the connectives constructions. We also show that purity may fail with $\multimap$, and only a weaker form called *quasi-purity* is always preserved.

▶ **Definition 16.** $\mathbf{B}$ is **regular** if the following conditions are satisfied:
- for all $\mathfrak{d} \in |B|$ and all path s of $\mathfrak{d}$, $s \in V_{\mathbf{B}}$,
- for all $\mathfrak{d} \in |B^\perp|$ and all path s of $\mathfrak{d}$, $s \in V_{\mathbf{B}^\perp}$,
- The sets $V_{\mathbf{B}}$ and $V_{\mathbf{B}^\perp}$ are stable under shuffle.

where the operation of **shuffle** ($\sqcup\!\sqcup$) on paths corresponds to an interleaving of actions respecting alternation of polarities, and is defined below.

Let $s\restriction s'$ refer to the subsequence of s containing only the actions that occur in s'. Let s and t be paths of same polarity, let S and T be sets of paths of same polarity. We define:
- $s \sqcup\!\sqcup t = \{u \text{ path formed with actions from } s \text{ and } t \mid u\restriction s = s \text{ and } u\restriction t = t\}$ if s, t negative,
- $s \sqcup\!\sqcup t = \{\kappa^+ u \text{ path} \mid u \in s' \sqcup\!\sqcup t'\}$ if $s = \kappa^+ s'$ and $t = \kappa^+ t'$ positive with same first action,
- $S \sqcup\!\sqcup T = \{u \text{ path} \mid \exists s \in S, \exists t \in T \text{ such that } s \sqcup\!\sqcup t \text{ is defined and } u \in s \sqcup\!\sqcup t\}$,

In fact, a behaviour $\mathbf{B}$ is regular if every path formed with actions of the incarnation of $\mathbf{B}$, even mixed up, is a visitable path of $\mathbf{B}$, and similarly for $\mathbf{B}^\perp$. Remark that regularity is a property of both a behaviour and its orthogonal since the definition is symmetrical: $\mathbf{B}$ is regular if and only if $\mathbf{B}^\perp$ is regular.

▶ **Definition 17.** A behaviour $\mathbf{B}$ is **pure** if every ✠-ended path $s✠ \in V_{\mathbf{B}}$ is **extensible**, i.e., there exists a proper positive action κ^+ such that $s\kappa^+ \in V_{\mathbf{B}}$.

Purity ensures that when an interaction encounters ✠, this does not correspond to a real error but rather to a partial computation, as it is possible to continue this interaction. Note that daimons are necessarily present in all behaviours since the converse property is always true: if $s\kappa^+ \in V_{\mathbf{B}}$ then $s✠ \in V_{\mathbf{B}}$.

▶ **Proposition 18.** *Regularity is stable under* $\downarrow$, $\uparrow$, $\oplus$, $\otimes$ *and* $\multimap$.

▶ **Proposition 19.** *Purity is stable under* $\downarrow$, $\uparrow$, $\oplus$ *and* $\otimes$.

Unfortunately, when $\mathbf{N}$ and $\mathbf{P}$ are pure, $\mathbf{N} \multimap \mathbf{P}$ is not necessarily pure, even under regularity assumption. However, a weaker form of purity holds for $\mathbf{N} \multimap \mathbf{P}$.

▶ **Definition 20.** A behaviour $\mathbf{B}$ is **quasi-pure** if all the ✠-ended *well-bracketed* paths in $V_{\mathbf{B}}$ are extensible.

We recall that a path s is **well-bracketed** if, for every justified action κ in s, when we write $s = s_0\kappa' s_1 \kappa s_2$ where κ' justifies κ, all the actions in s_1 are hereditarily justified by κ'.

▶ **Proposition 21.** *If* $\mathbf{N}$ *and* $\mathbf{P}$ *are quasi-pure and regular then* $\mathbf{N} \multimap \mathbf{P}$ *is quasi-pure.*

4 Inductive Data Types

Some important contributions are presented in this section. We interpret inductive data types as positive behaviours, and we prove an internal completeness result allowing us to make explicit the structure of fixed points. Regularity and purity of data follows.

Abusively, we denote the positive behaviour $\{✠\}$ by ✠ all along this section.

4.1 Inductive Data Types as Kleene Fixed Points

We define the *data patterns* via a type language and interpret them as behaviours, in particular μ is interpreted as a least fixed point. *Data behaviours* are the interpretation of *steady* data patterns.

Suppose given a countably infinite set $\mathcal{V}$ of second-order variables: $X, Y, \dots \in \mathcal{V}$. Let $\mathcal{S}' = \mathcal{S} \setminus \{▲, \pi_1, \pi_2, \wp\}$ and define the set of **constants** $\mathrm{Const} = \{\mathbf{C}_a \mid a \in \mathcal{S}'\}$ which contains a behaviour $\mathbf{C}_a = \{x_0|\overline{a}\langle\overrightarrow{\Omega^-}\rangle\}^{\perp\perp}$ (where $\Omega^- := \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\Omega$) for each $a \in \mathcal{S}'$, i.e., such that a is not the name of a connective. Remark that $V_{\mathbf{C}_a} = \{✠\,,\ x_0|\overline{a}\langle\overrightarrow{x}\rangle\}$, thus $\mathbf{C}_a$ is regular and pure.

▶ **Definition 22.** The set $\mathcal{P}$ of **data patterns** is generated by the inductive grammar:

$$A, B ::= X \in \mathcal{V} \quad | \quad a \in \mathcal{S}' \quad | \quad A \oplus^+ B \quad | \quad A \otimes^+ B \quad | \quad \mu X.A$$

The set of free variables of a data pattern $A \in \mathcal{P}$ is denoted by $\mathrm{FV}(A)$.

▶ **Example 23.** Let $b, n, l, t \in \mathcal{S}'$ and $X \in \mathcal{V}$. The data types given as example in the introduction can be written in the language of data patterns as follows:

$$\mathbb{B}\mathrm{ool} = b \oplus^+ b \qquad \mathbb{N}\mathrm{at} = \mu X.(n \oplus^+ X) \qquad \mathbb{L}\mathrm{ist}_A = \mu X.(l \oplus^+ (A \otimes^+ X))$$
$$\mathbb{T}\mathrm{ree}_A = \mu X.(t \oplus^+ (A \otimes^+ \mathbb{L}\mathrm{ist}_X)) = \mu X.(t \oplus^+ (A \otimes^+ \mu Y.(l \oplus^+ (X \otimes^+ Y))))$$

Let $\mathcal{B}^+$ be the set of positive behaviours. Given a data pattern $A \in \mathcal{P}$ and an environment σ, i.e., a function that maps free variables to positive behaviours, the interpretation of A in the environment σ, written $[\![A]\!]^\sigma$, is the positive behaviour defined by:

$$[\![X]\!]^\sigma = \sigma(X) \qquad [\![A \oplus^+ B]\!]^\sigma = (\uparrow[\![A]\!]^\sigma) \oplus (\uparrow[\![B]\!]^\sigma)$$
$$[\![a]\!]^\sigma = \mathbf{C}_a \qquad [\![A \otimes^+ B]\!]^\sigma = (\uparrow[\![A]\!]^\sigma) \otimes (\uparrow[\![B]\!]^\sigma)$$
$$[\![\mu X.A]\!]^\sigma = \mathrm{lfp}(\phi^A_\sigma)$$

where lfp stands for the least fixed point, and the function $\phi_\sigma^A : \mathcal{B}^+ \to \mathcal{B}^+, \mathbf{P} \mapsto [\![A]\!]^{\sigma, X \mapsto \mathbf{P}}$ is well defined and has a least fixed point by Knaster-Tarski fixed point theorem, as shown by Baelde, Doumane and Saurin [2]. Abusively we may write $\oplus^+$ and $\otimes^+$, instead of $(\uparrow\cdot) \oplus (\uparrow\cdot)$ and $(\uparrow\cdot) \otimes (\uparrow\cdot)$ respectively, for behaviours. We call an environment σ regular (resp. pure) if its image contains only regular (resp. pure) behaviours. The notation $\sigma, X \mapsto \mathbf{P}$ stands for the environment σ where the image of X has been changed to $\mathbf{P}$.

In order to understand the structure of fixed point behaviours that interpret the data patterns of the form $\mu X.A$, we need a constructive approach, thus Kleene fixed point theorem is best suited than Knaster-Tarski. We now prove that we can apply this theorem.

Recall the following definitions and theorem. A partial order is a **complete partial order** (CPO) if each directed subset has a supremum, and there exists a smallest element, written $\bot$. A function $f : E \to F$ between two CPOs is **Scott-continuous** (or simply continuous) if for every directed subset $D \subseteq E$ we have $\bigvee_{x \in D} f(x) = f(\bigvee_{x \in D} x)$.

▶ **Theorem 24** (Kleene fixed point theorem). *Let L be a CPO and let $f : L \to L$ be Scott-continuous. The function f has a least fixed point, defined by*

$$\mathrm{lfp}(f) = \bigvee_{n \in \mathbb{N}} f^n(\bot)$$

The set $\mathcal{B}^+$ ordered by $\subseteq$ is a CPO, with least element $\maltese$; indeed, given a subset $\mathbb{P} \subseteq \mathcal{B}^+$, it is directed and we have $\bigvee \mathbb{P} = (\bigcup \mathbb{P})^{\bot\bot}$. Hence next proposition proves that we can apply the theorem.

▶ **Proposition 25.** *Given a data pattern $A \in \mathcal{P}$, a variable $X \in \mathcal{V}$ and an environment $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$, the function ϕ_σ^A is Scott-continuous.*

▶ **Corollary 26.** *For every $A \in \mathcal{P}$, $X \in \mathcal{V}$ and $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$,*

$$[\![\mu X.A]\!]^\sigma = \bigvee_{n \in \mathbb{N}} (\phi_\sigma^A)^n(\maltese) = (\bigcup_{n \in \mathbb{N}} (\phi_\sigma^A)^n(\maltese))^{\bot\bot}.$$

This result gives an explicit formulation for least fixed points. However, the $\bot\bot$-closure might add new designs which were not in the union, making it difficult to know the exact content of such a behaviour. The point of next subsection will be to give an internal completeness result proving that the closure is actually not necessary.

Let us finish this subsection by defining a restricted set of data patterns so as to exclude the degenerate ones. Consider for example $\mathbb{L}ist_A{}' = \mu X.(A \otimes^+ X)$, a variant of $\mathbb{L}ist_A$ (see Example 23) which misses the base case. It is degenerate in the sense that the base element, here the empty list, is interpreted as the design $\maltese$. This is problematic: an interaction going through a whole list will end with an error, making it impossible to explore a pair of lists for example. The pattern $\mathbb{N}at' = \mu X.X$ is even worse since $[\![\mathbb{N}at']\!] = \maltese$. The point of steady data patterns is to ensure the existence of a basis; this will be formalised in Lemma 37.

▶ **Definition 27.** The set of **steady** data patterns is the smallest subset $\mathcal{P}^s \subseteq \mathcal{P}$ such that:

- $\mathcal{S}' \subseteq \mathcal{P}^s$
- If $A \in \mathcal{P}^s$ and B is such that $[\![B]\!]^\sigma$ is pure if σ is pure, then $A \oplus^+ B \in \mathcal{P}^s$ and $B \oplus^+ A \in \mathcal{P}^s$
- If $A \in \mathcal{P}^s$ and $B \in \mathcal{P}^s$ then $A \otimes^+ B \in \mathcal{P}^s$
- If $A \in \mathcal{P}^s$ then $\mu X.A \in \mathcal{P}^s$

The condition on B in the case of $\oplus^+$ admits data patterns which are not steady, possibly with free variables, but ensuring the preservation of purity, i.e., type safety; the basis will

come from side A. We will prove (§ 4.3) that behaviours interpreting steady data patterns are pure, thus in particular a data pattern of the form $\mu X.A$ is steady if the free variables of A all appear on the same side of a $\oplus^+$ and under the scope of no other μ (since purity is stable under $\downarrow, \uparrow, \oplus, \otimes$). We claim that steady data patterns can represent every type of finite data.

▶ **Definition 28.** A **data behaviour** is the interpretation of a closed steady data pattern.

4.2 Internal Completeness for Infinite Union

Our main result is an internal completeness theorem, stating that an infinite union of *simple* regular behaviours with increasingly large incarnations is a behaviour: $\perp\!\!\perp$-closure is useless.

▶ **Definition 29.**

- A **slice** is a design in which all negative subdesigns are either Ω^- or of the form $a(\overrightarrow{x}).\mathfrak{p}_a$, i.e., at most unary branching. $\mathfrak{c}$ is a **slice of** $\mathfrak{d}$ if $\mathfrak{c}$ is a slice and $\mathfrak{c} \sqsubseteq \mathfrak{d}$. A slice $\mathfrak{c}$ of $\mathfrak{d}$ is **maximal** if for any slice $\mathfrak{c}'$ of $\mathfrak{d}$ such that $\mathfrak{c} \sqsubseteq \mathfrak{c}'$, we have $\mathfrak{c} = \mathfrak{c}'$.
- A behaviour $\mathbf{B}$ is **simple** if for every design $\mathfrak{d} \in |\mathbf{B}|$:
  1. $\mathfrak{d}$ has a finite number of maximal slices, and
  2. every positive action of $\mathfrak{d}$ is justified by the immediate previous negative action.

Condition (2) of simplicity ensures that, given $\mathfrak{d} \in |\mathbf{B}|$ and a slice $\mathfrak{c} \sqsubseteq \mathfrak{d}$, one can find a path of $\mathfrak{c}$ containing all the positive proper actions of $\mathfrak{c}$ until a given depth; thus by condition (1), there exists $k \in \mathbb{N}$ depending only on $\mathfrak{d}$ such that k paths can do the same in $\mathfrak{d}$.

Now suppose $(\mathbf{A}_n)_{n\in\mathbb{N}}$ is an infinite sequence of simple regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$ (in particular we have $\mathbf{A}_n \subseteq \mathbf{A}_{n+1}$).

▶ **Theorem 30.** *The set $\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ is a behaviour.*

A union of behaviours is not a behaviour in general. In particular, counterexamples are easily found if releasing either the inclusion of incarnations or the simplicity condition. Moreover, our proof for this theorem relies strongly on regularity. Under the same hypotheses we can prove $V_{\bigcup_{n\in\mathbb{N}} \mathbf{A}_n} = \bigcup_{n\in\mathbb{N}} V_{\mathbf{A}_n}$ and $|\bigcup_{n\in\mathbb{N}} \mathbf{A}_n| = \bigcup_{n\in\mathbb{N}} |\mathbf{A}_n|$, hence the following corollary.

▶ **Corollary 31.**

- *$\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ is simple and regular;*
- *if moreover all the $\mathbf{A}_n$ are pure then $\bigcup_{n\in\mathbb{N}} \mathbf{A}_n$ is pure.*

4.3 Regularity and Purity of Data

The goal of this subsection is to show that the interpretation of data patterns of the form $\mu X.A$ can be expressed as an infinite union of behaviours $(\mathbf{A}_n)_{n\in\mathbb{N}}$ satisfying the hypotheses of Theorem 30, in order to deduce regularity and purity. We will call an environment σ simple if its image contains only simple behaviours.

▶ **Lemma 32.** *For all $A \in \mathcal{P}$, $X \in \mathcal{V}$, $\sigma : \mathrm{FV}(A) \setminus \{X\} \to \mathcal{B}^+$ and $n \in \mathbb{N}$ we have*

$$|(\phi^A_\sigma)^n(\maltese)| \subseteq |(\phi^A_\sigma)^{n+1}(\maltese)|\,.$$

▶ **Proposition 33.** *For all $A \in \mathcal{P}$ and simple regular environment σ, $[\![A]\!]^\sigma$ is simple regular.*

Proof. By induction on data patterns. If $A = X$ or $A = a$ the conclusion is immediate. If $A = A_1 \oplus^+ A_2$ or $A = A_1 \otimes^+ A_2$ then regularity comes from Proposition 18, and simplicity is easy since the structure of the designs in $[\![A]\!]^\sigma$ is given by internal completeness for the logical connectives (Theorem 8). So suppose $A = \mu X.A_0$. By induction hypothesis, for every simple regular behaviour $\mathbf{P} \in \mathcal{B}^+$ we have $\phi_\sigma^{A_0}(\mathbf{P}) = [\![A_0]\!]^{\sigma, X \mapsto \mathbf{P}}$ simple regular. From this, it is straightforward to show by induction that for every $n \in \mathbb{N}$, $(\phi_\sigma^{A_0})^n(\maltese)$ is simple regular. Moreover, for every $n \in \mathbb{N}$ we have $|(\phi_\sigma^{A_0})^n(\maltese)| \subseteq |(\phi_\sigma^{A_0})^{n+1}(\maltese)|$ by Lemma 32, thus by Corollary 26 and Theorem 30, $[\![\mu X.A_0]\!]^\sigma = \bigvee_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\maltese) = (\bigcup_{n\in\mathbb{N}} (\phi_\sigma^{A_0})^n(\maltese))^{\perp\perp} = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^{A_0})^n(\maltese)$. Consequently, by Corollary 31, $[\![\mu X.A_0]\!]^\sigma$ is simple regular. ◀

Remark that we have proved at the same time, using Theorem 30, that behaviours interpreting data patterns $\mu X.A$ admit an explicit construction:

▶ **Proposition 34.** *If $A \in \mathcal{P}$, $X \in \mathcal{V}$, and $\sigma : \mathrm{FV}(A) \setminus X \to \mathcal{B}^+$ is simple regular,*

$$[\![\mu X.A]\!]^\sigma = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\maltese)$$

▶ **Corollary 35.** *Data behaviours are regular.*

We now move on to proving purity. The proof that the interpretation of a steady data pattern A is pure relies on the existence of a basis for A (Lemma 37). Let us first widen (to $\maltese$-free paths) and express in a different way (for $\maltese$-ended paths) the notion of extensible visitable path.

▶ **Definition 36.** Let $\mathbf{B}$ be a behaviour.

- A $\maltese$-free path $s \in V_{\mathbf{B}}$ is **extensible** if there exists $t \in V_{\mathbf{B}}$ of which s is a strict prefix.
- A $\maltese$-ended path $s\maltese \in V_{\mathbf{B}}$ is **extensible** if there exists a positive action κ^+ and $t \in V_{\mathbf{B}}$ of which $s\kappa^+$ is a prefix.

Write $V_{\mathbf{B}}^{max}$ for the set of maximal, i.e., non extensible, visitable paths of $\mathbf{B}$.

▶ **Lemma 37.** *Every steady data pattern $A \in \mathcal{P}^s$ has a basis, i.e., a simple regular behaviour $\mathbf{B}$ such that for all simple regular environment σ we have*

- $\mathbf{B} \subseteq [\![A]\!]^\sigma$,
- *for every path $s \in V_{\mathbf{B}}$, there exists $t \in V_{\mathbf{B}}^{max}$ $\maltese$-free extending s (in particular $\mathbf{B}$ pure),*
- $V_{\mathbf{B}}^{max} \subseteq V_{[\![A]\!]^\sigma}^{max}$.

Proof (Idea). If $A = a$, a basis is $\mathbf{C}_a$. If $A = A_1 \oplus^+ A_2$, and A_i is steady with basis $\mathbf{B}_i$, then $\otimes_i \uparrow \mathbf{B}_i := \iota_i \langle \uparrow \mathbf{B}_i \rangle$ is a basis for A. If $A = A_1 \otimes^+ A_2$, a basis is $\mathbf{B}_1 \otimes^+ \mathbf{B}_2$ where $\mathbf{B}_1$ and $\mathbf{B}_2$ are basis of A_1 and A_2 respectively. If $A = \mu X.A_0$, its basis is the same as A_0. ◀

▶ **Proposition 38.** *If $A \in \mathcal{P}^s$ of basis $\mathbf{B}$, $X \in \mathcal{V}$, and $\sigma : \mathrm{FV}(A) \setminus X \to \mathcal{B}^+$ simple regular,*

$$[\![\mu X.A]\!]^\sigma = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\mathbf{B})$$

Proof. Since $\mathbf{B}$ is a basis for A we have $\maltese \subseteq \mathbf{B} \subseteq [\![A]\!]^{\sigma, X \to \maltese} = \phi_\sigma^A(\maltese)$. The Scott-continuity of the function ϕ_σ^A implies that it is increasing, thus $(\phi_\sigma^A)^n(\maltese) \subseteq (\phi_\sigma^A)^n(\mathbf{B}) \subseteq (\phi_\sigma^A)^{n+1}(\maltese)$ for all $n \in \mathbb{N}$. Hence $[\![A]\!]^\sigma = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\maltese) = \bigcup_{n\in\mathbb{N}} (\phi_\sigma^A)^n(\mathbf{B})$. ◀

▶ **Proposition 39.** *For all $A \in \mathcal{P}^s$ and simple regular pure environment σ, $[\![A]\!]^\sigma$ is pure.*

Proof. By induction on A. The base cases are immediate and the connective cases are solved using Proposition 19. Suppose now $A = \mu X.A_0$, where A_0 is steady with basis $\mathbf{B}_0$. We have $[\![A]\!]^\sigma = \bigcup_{n\in\mathbb{N}}(\phi_\sigma^{A_0})^n(\mathbf{B}_0)$ by Proposition 38, let us prove it satisfies the hypotheses needed to apply Corollary 31(2). By induction hypothesis and Proposition 33, for every simple, regular and pure behaviour $\mathbf{P} \in \mathcal{B}^+$ we have $\phi_\sigma^{A_0}(\mathbf{P}) = [\![A_0]\!]^{\sigma,X\mapsto\mathbf{P}}$ simple, regular and pure, hence it is easy to show by induction that for every $n \in \mathbb{N}$, $(\phi_\sigma^{A_0})^n(\mathbf{B}_0)$ is as well. Moreover, for every $n \in \mathbb{N}$ we prove that $|(\phi_\sigma^{A_0})^n(\mathbf{B}_0)| \subseteq |(\phi_\sigma^{A_0})^{n+1}(\mathbf{B}_0)|$ similarly to Lemma 32, replacing ✠ by the basis $\mathbf{B}_0$. Finally, by Corollary 31, $[\![A]\!]^\sigma$ is pure. ◀

▶ **Corollary 40.** *Data behaviours are pure.*

▶ Remark. Although here the focus is on the interpretation of data patterns, we should say a word about the interpretation of (polarised) μMALL formulas, which are a bit more general. These formulas are generated by:

$$\begin{array}{rcl} P,Q & ::= & X_P \mid X_N^\perp \mid 1 \mid 0 \mid M \oplus N \mid M \otimes N \mid {\downarrow}N \mid \mu X.P \\ M,N & ::= & P^\perp \end{array}$$

where the usual involutive negation hides the negative connectives and constants, through the dualities $1/\bot$, $0/\top$, $\oplus/\&$, $\otimes/\parr$, $\downarrow/\uparrow$, μ/ν . The interpretation as ludics behaviours, given in [2], is as follows: 1 is interpreted as a constant behaviour $\mathbf{C}_a$, 0 is the daimon ✠, the positive connectives match their ludics counterparts, μ is interpreted as the least fixed point of a function ϕ_σ^A similarly to data patterns, and the negation corresponds to the orthogonal. Since in ludics constants and ✠ are regular, and since regularity is preserved by the connectives (Proposition 18) and by orthogonality, the only thing we need in order to prove that all the behaviours interpreting μMALL formulas are regular is a generalisation of regularity stability under fixed points (for now we only have it in our particular case: Corollary 31 together with Proposition 34).

Note however that interpretations of μMALL formulas are not all pure. Indeed, as we will see in next section, orthogonality (introduced through the connective $\multimap$) does not preserve purity in general.

5 Functional Types

In this section we define *functional behaviours* which combine data behaviours with the connective $\multimap$. A behaviour of the form $\mathbf{N} \multimap \mathbf{P}$ is the set of designs such that, when interacting with a design of type $\mathbf{N}$, outputs a design of type $\mathbf{P}$; this is exactly the meaning of its definition $\mathbf{N} \multimap \mathbf{P} := (\mathbf{N} \otimes \mathbf{P}^\perp)^\perp$. We prove that some particular higher-order functional types – where functions are taken as arguments, typically $(A \multimap B) \multimap C$ – are exactly those who fail at being pure, and we interpret this result from a computational point of view.

5.1 Where Impurity Arises

We have proved that data behaviours are regular and pure. However, if we introduce functional behaviours with the connective $\multimap$, purity does not hold in general. Proposition 42 indicates that a weaker property, quasi-purity, holds for functional types, and Proposition 43 identifies exactly the cases where purity fails.

Let us write $\mathcal{D}$ for the set of data behaviours.

▶ **Definition 41.** A **functional behaviour** is a behaviour inductively generated by the grammar below, where $\mathbf{P} \multimap^+ \mathbf{Q}$ stands for $\downarrow((\uparrow\mathbf{P}) \multimap \mathbf{Q})$.

$$\mathbf{P}, \mathbf{Q} ::= \mathbf{P}_0 \in \mathcal{D} \quad | \quad \mathbf{P} \oplus^+ \mathbf{Q} \quad | \quad \mathbf{P} \otimes^+ \mathbf{Q} \quad | \quad \mathbf{P} \multimap^+ \mathbf{Q}\,.$$

From Propositions 18, 19 and 21 we easily deduce the following result.

▶ **Proposition 42.** *Functional behaviours are regular and quasi-pure.*

For next proposition, consider **contexts** defined inductively as follows (where $\mathbf{P}$ is a functional behaviour):

$$\mathcal{C} ::= [\,] \quad | \quad \mathcal{C} \oplus^+ \mathbf{P} \quad | \quad \mathbf{P} \oplus^+ \mathcal{C} \quad | \quad \mathcal{C} \otimes^+ \mathbf{P} \quad | \quad \mathbf{P} \otimes^+ \mathcal{C} \quad | \quad \mathbf{P} \multimap^+ \mathcal{C}\,.$$

▶ **Proposition 43.** *A functional behaviour $\mathbf{P}$ is impure if and only if there exist contexts $\mathcal{C}_1, \mathcal{C}_2$ and functional behaviours $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{R}$ with $\mathbf{R} \notin$ Const such that*

$$\mathbf{P} = \mathcal{C}_1[\; \mathcal{C}_2[\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2] \multimap^+ \mathbf{R} \;]\,.$$

5.2 Example and Discussion

Proposition 43 states that a functional behaviour which takes functions as argument is not pure: some of its visitable paths end with a daimon ✠, and there is no possibility to extend them. In terms of proof-search, playing the daimon is like giving up; on a computational point of view, the daimon appearing at the end of an interaction expresses the sudden interruption of the computation. In order to understand why such an interruption can occur in the specific case of higher-order functions, consider the following example which illustrates the proposition.

▶ **Example 44.** Let $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{1}$ be functional behaviours, with $\mathbf{1} \in$ Const. Define $\mathbf{Bool} = \mathbf{1} \oplus^+ \mathbf{1}$ and consider the behaviour $\mathbf{P} = (\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2) \multimap^+ \mathbf{Bool}$: this is a type of functions which take a function as argument and output a boolean. Let $\alpha_1, \alpha_2, \beta$ be respectively the first positive action of the designs of $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{1}$. It is possible to exhibit a design $\mathfrak{p} \in \mathbf{P}$ and a design $\mathfrak{n} \in \mathbf{P}^\perp$ such that the visitable path $s = \langle \mathfrak{p} \leftarrow \mathfrak{n} \rangle$ is ✠-ended and maximal in $V_{\mathbf{P}}$, in other words s is a witness of the impurity of $\mathbf{P}$. The path s contains the actions α_1 and $\overline{\alpha_2}$ in such a way that it cannot be extended with β without breaking the P-visibility condition, and there is no other available action in designs of $\mathbf{P}$ to extend it. Reproducing the designs $\mathfrak{p}$ and $\mathfrak{n}$ and the path s here would be of little interest since those objects are too large to be easily readable (s visits the entire design $\mathfrak{p}$, which contains 11 actions). We however give an intuition in the style of game semantics: Fig. 2 represents s as a legal play in a strategy of type $\mathbf{P} = (\mathbf{Q}_1 \multimap^+ \mathbf{Q}_2) \multimap^+ \mathbf{Bool}$ (note that only one "side" $\oplus_1\uparrow\mathbf{1}$ of **Bool** is represented, corresponding for example to `True`, because we cannot play in both sides). This analogy is informal, it should stand as an intuition rather than as a precise correspondence with ludics; for instance, and contrary to the way it is presented in game semantics, the questions are asked on the connectives, while the answers are given in the sub-types of $\mathbf{P}$. On the right are given the actions in s corresponding to the moves played. The important thing to remark is the following: if a move b corresponding to action β were played instead of ✠ at the end of this play, it would break the P-visibility of the strategy, since this move would be justified by move $q_\uparrow$.

The computational interpretation of the ✠-ended interaction between $\mathfrak{p}$ and $\mathfrak{n}$ is the following: a program p of type $\mathbf{P}$ launches a child process p' to compute the argument of type $\mathbf{Q}_1 \to \mathbf{Q}_2$, but p starts to give a result in **Bool** before the execution of p' terminates,

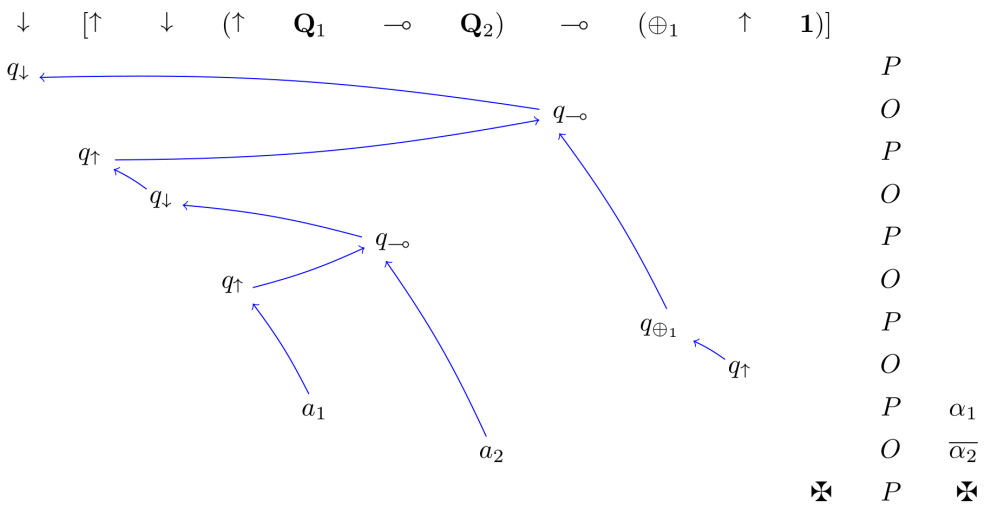


Figure 2 Representation of path s from Example 44 in the style of a legal play.

leading to a situation where p cannot compute the whole data in **Bool**. The interaction outputs ✠, i.e., the answer given in **Bool** by p is incomplete.

Moreover by Proposition 42 functional behaviours are quasi-pure, therefore the maximal ✠-ended visitable paths are necessarily not well-bracketed. This is indeed the case of s: remark for example that the move $q_{\oplus_1}$ appears between a_1 and its justification $q_\uparrow$ in the sequence, but $q_{\oplus_1}$ is not hereditarily justified by $q_\uparrow$. In HO games, well-bracketedness is a well studied notion, and relaxing it introduces control operators in program. If we extend such an argument to ludics, this would mean that the appearance of ✠ in the execution of higher-order functions can only happen in the case of programs with control operators such as *jumps*, i.e. programs which are not purely functional.

6 Conclusion

This article is a contribution to the exploration of the behaviours of linear ludics in a computational perspective. Our focus is on the behaviours representing data types and functional types. Inductive data types are interpreted using the logical connectives constructions and a least fixed point operation. Adopting a constructive approach, we provide an internal completeness result for fixed points, which unveils the structure of data behaviours. This leads us to proving that such behaviours are regular – the key notion for the characterisation of MALL in ludics – and pure – that is, type safe. But behaviours interpreting types of functions taking functions as argument are impure; for well-bracketed interactions, corresponding to the evaluation of purely functional programs, safety is however guaranteed.

Further Work. Two directions for future research arise naturally:

- Extending our study to greatest fixed points $\nu X.A$, i.e., coinduction, is the next objective. Knaster–Tarski ensures that such greatest fixed point behaviours exist [2], but Kleene fixed point theorem does not apply here, hence we cannot find an explicit form for coinductive behaviours the same way we did for the inductive ones. However it is intuitively clear

that, compared to least fixed points, greatest ones add the infinite "limit" designs in (the incarnation of) behaviours. For example, if $\mathbb{N}at_\omega = \nu X.(1 \oplus X)$ then we should have $|[\![\mathbb{N}at_\omega]\!]| = |[\![\mathbb{N}at]\!]| \cup \{\mathfrak{d}_\omega\}$ where $\mathfrak{d}_\omega = \mathrm{succ}(\mathfrak{d}_\omega) = x_0|\iota_2\langle\uparrow(x).{\mathfrak{d}_\omega}^x\rangle$.

- Another direction would be to get a complete characterisation of μMALL in ludics, by proving that a behaviour is regular – and possibly satisfying a supplementary condition – if and only if it is the denotation of a μMALL formula.

Acknowledgements. I thank Claudia Faggian, Christophe Fouqueré, Thomas Seiller and the anonymous referees for their wise and helpful comments.

References

1 David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Logic*, 13(1):2:1–2:44, January 2012. doi:10.1145/2071368.2071370.

2 David Baelde, Amina Doumane, and Alexis Saurin. Least and greatest fixed points in ludics. In Stephan Kreuzer, editor, *Proceedings of the 24th Annual EACSL Conference on Computer Science Logic (CSL'15)*, pages 549–566, Berlin, Germany, September 2015. doi:10.4230/LIPIcs.CSL.2015.549.

3 Michele Basaldella and Claudia Faggian. Ludics with repetitions (exponentials, interactive types and completeness). *Logical Methods in Computer Science*, 7(2), 2011.

4 Pierre Clairambault. Least and greatest fixpoints in game semantics. In *Foundations of Software Science and Computational Structures, 12th International Conference (FOSSACS 2009). Proceedings*, pages 16–31, 2009.

5 Pierre-Louis Curien. Introduction to linear logic and ludics, part II. *CoRR*, abs/cs/0501039, 2005.

6 Claudia Faggian and Martin Hyland. Designs, disputes and strategies. In *CSL*, pages 442–457, 2002.

7 Christophe Fouqueré and Myriam Quatrini. Incarnation in ludics and maximal cliques of paths. *Logical Methods in Computer Sciences*, 9(4), 2013.

8 Christophe Fouqueré and Myriam Quatrini. Study of behaviours via visitable paths. *CoRR*, abs/1403.3772v3, 2016. URL: https://arxiv.org/abs/1403.3772v3.

9 Jean-Yves Girard. Geometry of interaction. I. Interpretation of system f. In *Proc. Logic Colloquium 1988*, pages 221–260, North-Holland, Amsterdam, 1989.

10 Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

11 Martin Hyland and Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.

12 Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009. URL: https://hal.archives-ouvertes.fr/hal-00154500.

13 Paul-André Melliès and Jerome Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 82–91, 2005. doi:10.1109/LICS.2005.42.

14 Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

15 Eugenia Sironi. *Types in Ludics*. PhD thesis, Aix-Marseille Université, January 2015.

16 Kazushige Terui. Computational ludics. *Theor. Comput. Sci.*, 412(20):2048–2071, 2011.

A Technical Appendix

This appendix presents the proof of Theorem 30, which requires first some preliminaries.

A.1 Observational Ordering and Monotonicity

We consider the **observational ordering** $\preceq$ over designs: $\mathfrak{d}' \preceq \mathfrak{d}$ if $\mathfrak{d}$ can be obtained from $\mathfrak{d}'$ by substituting:

- positive subdesigns for some occurrences of Ω.
- $\maltese$ for some positive subdesigns.

Remark in particular that for all positive designs $\mathfrak{p}$ and $\mathfrak{p}'$, we have $\Omega \preceq \mathfrak{p} \preceq \maltese$, and if $\mathfrak{p} \sqsubseteq \mathfrak{p}'$ then $\mathfrak{p} \preceq \mathfrak{p}'$. We can now state the *monotonicity* theorem, an important result of ludics. A proof of the theorem formulated in this form is found in [16].

▶ **Theorem 45** (Monotonicity)**.**

- *If* $\mathfrak{d} \preceq \mathfrak{e}$ *and* $\mathfrak{m} \preceq \mathfrak{n}$*, then* $\mathfrak{d}[\mathfrak{m}/x] \preceq \mathfrak{e}[\mathfrak{n}/x]$.
- *If* $\mathfrak{d} \preceq \mathfrak{e}$ *then* $(\!|\mathfrak{d}|\!) \preceq (\!|\mathfrak{e}|\!)$.

This means that the relation $\preceq$ compares the likelihood of convergence: if $\mathfrak{d} \perp \mathfrak{e}$ and $\mathfrak{d} \preceq \mathfrak{d}'$ then $\mathfrak{d}' \perp \mathfrak{e}$. In particular, if $\mathbf{B}$ is a behaviour, if $\mathfrak{d} \in \mathbf{B}$ and $\mathfrak{d} \preceq \mathfrak{d}'$ then $\mathfrak{d}' \in \mathbf{B}$.

Remark the following important fact: given a path s of some design $\mathfrak{d}$, there is a unique design maximal for $\preceq$ such that s is a path of it. Indeed, this design $\ulcorner s \urcorner^c$ is obtained from $\mathfrak{d}$ by replacing all positive subdesigns (possibly Ω) whose first positive action is not in s by $\maltese$. Note that, actually, the design $\ulcorner s \urcorner^c$ does not depend on $\mathfrak{d}$ but only on the path s.

▶ **Proposition 46.** *For every behaviour* $\mathbf{B}$*, if* $s \in V_{\mathbf{B}}$ *then* $\ulcorner s \urcorner^c \in \mathbf{B}$.

A.2 More on Paths

Let $\mathbf{B}$ be a behaviour.

▶ **Lemma 47.** *If* $\mathfrak{d} \in \mathbf{B}$ *and* $s \in V_{\mathbf{B}}$ *is a path of* $\mathfrak{d}$*, then* s *is a path of* $|\mathfrak{d}|$.

▶ **Lemma 48.** *Let* $s \in V_{\mathbf{B}}$*. For every positive-ended (resp. negative-ended) prefix* s' *of* s*, we have* $s' \in V_{\mathbf{B}}$ *(resp.* $s'\maltese \in V_{\mathbf{B}}$*).*

▶ **Lemma 49.** *Let* $s \in V_{\mathbf{B}}$*. For every prefix* $s'\kappa^-$ *of* s *and every* $\mathfrak{d} \in \mathbf{B}$ *such that* s' *is a path of* $\mathfrak{d}$*,* $s'\kappa^-$ *is a prefix of a path of* $\mathfrak{d}$.

A.3 An Alternative Definition of Regularity

Define the **anti-shuffle** ($\sqcap\!\!\sqcap$) as the dual operation of shuffle, that is:

- $s \sqcap\!\!\sqcap t = \widetilde{\tilde{s} \sqcup\!\!\sqcup \tilde{t}}$ if s and t are paths of same polarity;
- $S \sqcap\!\!\sqcap T = \widetilde{\tilde{S} \sqcup\!\!\sqcup \tilde{T}}$ if S and T are sets of paths of same polarity.

▶ **Definition 50.**

- A **trivial view** is an aj-sequence such that each proper action except the first one is justified by the immediate previous action. In other words, it is a view such that its dual is a view as well.

- The **trivial view of** an aj-sequence is defined inductively by:

$$\begin{aligned}
\langle \epsilon \rangle &= \epsilon && \text{empty sequence} \\
\langle s✠ \rangle &= \langle s \rangle ✠ && \\
\langle s\kappa \rangle &= \kappa && \text{if } \kappa \neq ✠ \text{ initial} \\
\langle s\kappa \rangle &= \langle s_0 \rangle \kappa && \text{if } \kappa \neq ✠ \text{ justified, where } s_0 \text{ prefix of } s \text{ ending on } just(\kappa)
\end{aligned}$$

  We also write $\langle \kappa \rangle_s$ (or even $\langle \kappa \rangle$) instead of $\langle s'\kappa \rangle$ when $s'\kappa$ is a prefix of s.
- **Trivial views of a design** $\mathfrak{d}$ are the trivial views of its paths (or of its views). In particular, ϵ is a trivial view of negative designs only.
- Trivial views of designs in $|\mathbf{B}|$ are called **trivial views of B**.

▶ **Lemma 51.**
1. *Every view is in the anti-shuffle of trivial views.*
2. *Every path is in the shuffle of views.*

▶ Remark. Following previous result, note that every view (resp. path) of a design $\mathfrak{d}$ is in the anti-shuffle of trivial views (resp. in the shuffle of views) of $\mathfrak{d}$.

▶ **Proposition 52.** $\mathbf{B}$ *is regular if and only if the following conditions hold:*
- *the positive-ended trivial views of* $\mathbf{B}$ *are visitable in* $\mathbf{B}$*,*
- $V_{\mathbf{B}}$ *and* $V_{\mathbf{B}^\perp}$ *are stable under* $\sqcup\!\sqcup$ *(i.e.,* $V_{\mathbf{B}}$ *is stable under* $\sqcup\!\sqcup$ *and* $\sqcap$*).*

A.4 Proof of Theorem 30

Before proving Theorem 30 we need some lemmas. Suppose $(\mathbf{A}_n)_{n\in\mathbb{N}}$ is an infinite sequence of regular behaviours such that for all $n \in \mathbb{N}$, $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$; the simplicity hypothesis is not needed for now. Let us note $\mathbf{A} = \bigcup_{n\in\mathbb{N}} \mathbf{A}_n$. Notice that the definition of visitable paths can harmlessly be extended to any set E of designs of same polarity, even if it is not a behaviour; the same applies to the definition of incarnation, provided that E satisfies the following: if $\mathfrak{d}, \mathfrak{e}_1, \mathfrak{e}_2 \in E$ are cut-free designs such that $\mathfrak{e}_1 \sqsubseteq \mathfrak{d}$ and $\mathfrak{e}_2 \sqsubseteq \mathfrak{d}$ then there exists $\mathfrak{e} \in E$ cut-free such that $\mathfrak{e} \sqsubseteq \mathfrak{e}_1$ and $\mathfrak{e} \sqsubseteq \mathfrak{e}_2$. In particular, as a union of behaviours, $\mathbf{A}$ satisfies this condition.

▶ **Lemma 53.**
1. $\forall n \in \mathbb{N}$, $V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}_{n+1}}$.
2. $V_{\bigcup_{n\in\mathbb{N}} \mathbf{A}_n} = \bigcup_{n\in\mathbb{N}} V_{\mathbf{A}_n}$.
3. $|\bigcup_{n\in\mathbb{N}} \mathbf{A}_n| = \bigcup_{n\in\mathbb{N}} |\mathbf{A}_n|$.

Proof.
1. Fix n and let $s \in V_{A_n}$. There exist $\mathfrak{d} \in |\mathbf{A}_n|$ such that s is a path of $\mathfrak{d}$. Since $|\mathbf{A}_n| \subseteq |\mathbf{A}_{n+1}|$ we have $\mathfrak{d} \in |\mathbf{A}_{n+1}|$, thus by regularity of $\mathbf{A}_{n+1}$, $s \in V_{A_{n+1}}$.
2. ($\subseteq$) Let $s \in V_{\mathbf{A}}$. There exist $n \in \mathbb{N}$ and $\mathfrak{d} \in |\mathbf{A}_n|$ such that s is a path of $\mathfrak{d}$. By regularity of $\mathbf{A}_n$ we have $s \in V_{\mathbf{A}_n}$.
   ($\supseteq$) Let $m \in \mathbb{N}$ and $s \in V_{\mathbf{A}_m}$. For all $n \geq m$, $V_{\mathbf{A}_m} \subseteq V_{\mathbf{A}_n}$ by previous item, thus $s \in V_{\mathbf{A}_n}$. Hence if we take $\mathfrak{e} = \overset{\sqcap\sqcap c}{\tilde{s}}$, we have $\mathfrak{e} \in {\mathbf{A}_n}^\perp$ for all $n \geq m$ by monotonicity. We deduce $\mathfrak{e} \in \bigcap_{n\geq m} {\mathbf{A}_n}^\perp = (\bigcup_{n\geq m} \mathbf{A}_n)^\perp = (\bigcup_{n\in\mathbb{N}} \mathbf{A}_n)^\perp = \mathbf{A}^\perp$. Let $\mathfrak{d} \in \mathbf{A}_m$ such that s is a path of $\mathfrak{d}$; we have $\mathfrak{d} \in \mathbf{A}$ and $\mathfrak{e} \in \mathbf{A}^\perp$, thus $\langle \mathfrak{d} \leftarrow \mathfrak{e} \rangle = s \in V_{\mathbf{A}}$.
3. ($\subseteq$) Let $\mathfrak{d}$ be cut-free and minimal for $\sqsubseteq$ in $\mathbf{A}$. There exists $m \in \mathbb{N}$ such that $\mathfrak{d} \in \mathbf{A}_m$. Thus $\mathfrak{d}$ is minimal for $\sqsubseteq$ in $\mathbf{A}_m$ otherwise it would not be minimal in $\mathbf{A}$, hence the result.
   ($\supseteq$) Let $m \in \mathbb{N}$, and let $\mathfrak{d} \in |\mathbf{A}_m|$. By hypothesis, $\mathfrak{d} \in |\mathbf{A}_n|$ for all $n \geq m$. Suppose $\mathfrak{d}$ is not in $|\mathbf{A}|$, so there exists $\mathfrak{d}' \in \mathbf{A}$ such that $\mathfrak{d}' \sqsubseteq \mathfrak{d}$ and $\mathfrak{d}' \neq \mathfrak{d}$. In this case, there exists $n \geq m$ such that $\mathfrak{d}' \in \mathbf{A}_n$, but this contradicts the fact that $\mathfrak{d} \in |\mathbf{A}_n|$. ◀

▶ **Lemma 54.** $V_{\bigcup_{n\in\mathbb{N}}\mathbf{A}_n} = \widetilde{V_{\bigcup_{n\in\mathbb{N}}\mathbf{A}_n{}^{\perp}}} = V_{(\bigcup_{n\in\mathbb{N}}\mathbf{A}_n)^{\perp\perp}}$.

Proof. In this proof we use the alternative definition of regularity (Proposition 52). We prove $V_{\mathbf{A}} = \widetilde{V_{\mathbf{A}^{\perp}}}$, and the result will follow from the fact that for any behaviour $\mathbf{B}$ (in particular if $\mathbf{B} = \mathbf{A}^{\perp\perp}$) we have $\widetilde{V_{\mathbf{B}^{\perp}}} = V_{\mathbf{B}}$. First note that the inclusion $V_{\mathbf{A}} \subseteq \widetilde{V_{\mathbf{A}^{\perp}}}$ is immediate.

Let $s \in V_{\mathbf{A}^{\perp}}$ and let us show that $\tilde{s} \in V_{\mathbf{A}}$. Let $\mathfrak{e} \in |\mathbf{A}^{\perp}|$ such that s is a path of $\mathfrak{e}$. By Lemma 51 and the remark following it, s is in the shuffle of anti-shuffles of trivial views $\approx_1, \dots, \approx_k$ of $\mathbf{A}^{\perp}$. For every $i \leq k$, suppose $\approx_i = \langle \kappa_i \rangle$; necessarily, there exists a design $\mathfrak{d}_i \in \mathbf{A}$ such that κ_i occurs in $\langle \mathfrak{e} \leftarrow \mathfrak{d}_i \rangle$, i.e., such that $\approx_i$ is a subsequence of $\langle \mathfrak{e} \leftarrow \mathfrak{d}_i \rangle$, otherwise $\mathfrak{e}$ would not be in the incarnation of $\mathbf{A}^{\perp}$ (it would not be minimal). Let n be big enough such that $\mathfrak{d}_1, \dots, \mathfrak{d}_k \in \mathbf{A}_n$, and note that in particular $\mathfrak{e} \in \mathbf{A}_n{}^{\perp}$. For all i, $\widetilde{\approx_i}$ is a trivial view of $|\mathfrak{d}_i|_{\mathbf{A}_n}$, thus it is a trivial view of $\mathbf{A}_n$. By regularity of $\mathbf{A}_n$ we have $\widetilde{\approx_i} \in V_{\mathbf{A}_n}$. Since $\tilde{s}$ is in the anti-shuffle of shuffles of $\widetilde{\approx_1}, \dots, \widetilde{\approx_k}$, we have $\tilde{s} \in V_{\mathbf{A}_n}$ using regularity again. Therefore $\tilde{s} \in V_{\mathbf{A}}$ by Lemma 53. ◀

▶ **Lemma 55.** $(\bigcup_{n\in\mathbb{N}}\mathbf{A}_n)^{\perp}$ *and* $(\bigcup_{n\in\mathbb{N}}\mathbf{A}_n)^{\perp\perp}$ *are regular.*

Proof. Let us show $\mathbf{A}^{\perp}$ is regular using the equivalent definition (Proposition 52).

- Let $\approx$ be a trivial view of $\mathbf{A}^{\perp}$. By a similar argument as in the proof above, there exists $n \in \mathbb{N}$ such that $\widetilde{\approx}$ is a trivial view of $\mathbf{A}_n$, thus $\widetilde{\approx} \in V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}}$. By Lemma 54 $\approx \in V_{\mathbf{A}^{\perp}}$.
- Let $s, t \in V_{\mathbf{A}^{\perp}}$. By Lemma 54, $\tilde{s}, \tilde{t} \in V_{\mathbf{A}}$. By Lemma 53(2), there exists $n \in \mathbb{N}$ such that $\tilde{s}, \tilde{t} \in V_{\mathbf{A}_n}$, thus by regularity of $\mathbf{A}_n$ we have $\tilde{s} \pitchfork \tilde{t}$, $\tilde{s} \sqcup \tilde{t} \subseteq V_{\mathbf{A}_n} \subseteq V_{\mathbf{A}}$, in other words $\widetilde{s \sqcup t}$, $\widetilde{s \pitchfork t} \subseteq V_{\mathbf{A}}$. By Lemma 54 we deduce $s \sqcup t$, $s \pitchfork t \subseteq V_{\mathbf{A}^{\perp}}$, hence $V_{\mathbf{A}^{\perp}}$ is stable under shuffle and anti-shuffle.

Finally $\mathbf{A}^{\perp}$ is regular. We deduce that $\mathbf{A}^{\perp\perp}$ is regular since regularity is stable under orthogonality. ◀

Let us introduce some more notions for next proof. An **∞-path** (resp. **∞-view**) is a finite or infinite sequence of actions satisfying all the conditions of the definition of path (resp. view) but the requirement of finiteness. In particular, a finite ∞-path (resp. ∞-view) is a path (resp. a view). An **∞-path** (resp. **∞-view**) **of** a design $\mathfrak{d}$ is such that any of its positive-ended prefix is a path (resp. a view) of $\mathfrak{d}$. We call **infinite chattering** a closed interaction which diverges because the computation never ends; note that infinite chattering occurs in the interaction between two atomic designs $\mathfrak{p}$ and $\mathfrak{n}$ if and only if there exists an infinite ∞-path s of $\mathfrak{p}$ such that $\tilde{s}$ is an ∞-path of $\mathfrak{n}$ (where, when s is infinite, $\tilde{s}$ is obtained from s by simply reversing the polarities of all the actions). Given an infinite ∞-path s, the design $\ulcorner s \urcorner^{\mathrm{c}}$ is constructed similarly to the case when s is finite (see §A.1).

For the proof of the theorem, suppose now that the behaviours $(\mathbf{A}_n,)_{n\in\mathbb{N}}$ are simple. Remark that the second condition of simplicity implies in particular that the dual of a path in a design of a simple behaviour is a view.

Proof of Theorem 30. We must show that $\mathbf{A}^{\perp\perp} \subseteq \mathbf{A}$ since the other inclusion is trivial. Remark the following: given designs $\mathfrak{d}$ and $\mathfrak{d}'$, if $\mathfrak{d} \in \mathbf{A}$ and $\mathfrak{d} \sqsubseteq \mathfrak{d}'$ then $\mathfrak{d}' \in \mathbf{A}$. Indeed, if $\mathfrak{d} \in \mathbf{A}$ then there exists $n \in \mathbb{N}$ such that $\mathfrak{d} \in \mathbf{A}_n$; if moreover $\mathfrak{d} \sqsubseteq \mathfrak{d}'$ then in particular $\mathfrak{d} \preceq \mathfrak{d}'$, and by monotonicity $\mathfrak{d}' \in \mathbf{A}_n$, hence $\mathfrak{d}' \in \mathbf{A}$. Thus it is sufficient to show $|\mathbf{A}^{\perp\perp}| \subseteq \mathbf{A}$ since for every $\mathfrak{d}' \in \mathbf{A}^{\perp\perp}$ we have $|\mathfrak{d}'| \in |\mathbf{A}^{\perp\perp}|$ and $|\mathfrak{d}'| \sqsubseteq \mathfrak{d}'$.

So let $\mathfrak{d} \in |\mathbf{A}^{\perp\perp}|$ and suppose $\mathfrak{d} \notin \mathbf{A}$. First note the following: by Lemmas 54 and 55, every path s of $\mathfrak{d}$ is in $V_{\mathbf{A}^{\perp\perp}} = V_{\mathbf{A}}$, thus there exists $\mathfrak{d}' \in |\mathbf{A}|$ containing s. We explore separately the possible cases, and show how they all lead to a contradiction.

If $\mathfrak{d}$ has an infinite number of maximal slices then:

- Either there exists a negative subdesign $\mathfrak{n} = \sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ of $\mathfrak{d}$ for which there is an infinity of names $a \in \mathcal{A}$ such that $\mathfrak{p}_a \neq \Omega$. In this case, let $\gtrsim$ be the view of $\mathfrak{d}$ such that for every action κ^- among the first ones of $\mathfrak{n}$, $\gtrsim\kappa^-$ is the prefix of a view of $\mathfrak{d}$. All such sequences $\gtrsim\kappa^-$ being prefixes of paths of $\mathfrak{d}$, we deduce by regularity of $\mathbf{A}^{\perp\perp}$ and using Lemma 48 that $\gtrsim\kappa^-\maltese \in V_{\mathbf{A}^{\perp\perp}}$. Let $\mathfrak{d}' \in |\mathbf{A}|$ be such that $\gtrsim$ is a view of $\mathfrak{d}'$. Since $\mathfrak{d}'$ is also in $\mathbf{A}^{\perp\perp}$, we deduce by Lemma 49 that for every action κ^- among the first ones of $\mathfrak{n}$, $\gtrsim\kappa^-$ is the prefix of a view of $\mathfrak{d}'$. Thus $\mathfrak{d}'$ has an infinite number of slices: contradiction.
- Or we can find an infinite ∞-view $\gtrsim = (\kappa_0^-)\kappa_1^+\kappa_1^-\kappa_2^+\kappa_1^-\kappa_3^+\kappa_3^- \dots$ of $\mathfrak{d}$ (the first action κ_0^- being optional depending on the polarity of $\mathfrak{d}$) satisfying the following: there is an infinity of $i \in \mathbb{N}$ such than κ_i^- is one of the first actions of a negative subdesign $\sum_{a\in\mathcal{S}} a(\overrightarrow{x^a}).\mathfrak{p}_a$ of $\mathfrak{d}$ with at least two names $a \in \mathcal{A}$ such that $\mathfrak{p}_a \neq \Omega$. Let $\gtrsim_i$ be the prefix of $\gtrsim$ ending on κ_i^+. There is no design $\mathfrak{d}' \in |\mathbf{A}|$ containing $\gtrsim$, indeed: in this case, for all i and all negative action κ^- such that $\gtrsim_i\kappa^-$ is a prefix of a view of $\mathfrak{d}$, $\gtrsim_i\kappa^-$ would be a prefix of a view of $\mathfrak{d}'$ by Lemma 49, thus $\mathfrak{d}'$ would have an infinite number of slices, which is impossible since the $\mathbf{A}_n$ are simple. Thus consider $\mathfrak{e} = \overset{\maltese\sim\dashv^c}{\gtrsim}$: since all the $\overset{\sim}{\gtrsim_i}$ are views of designs in $|\mathbf{A}| = \bigcup_{n\in\mathbb{N}} |\mathbf{A}_n|$ and since the $\mathbf{A}_n$ are simple, the sequences $\overset{\sim}{\gtrsim_i}$ are views, thus $\overset{\sim}{\gtrsim}$ is an ∞-view. Therefore an interaction between a design $\mathfrak{d}' \in \mathbf{A}$ and $\mathfrak{e}$ necessarily eventually converges by reaching a daimon of $\mathfrak{e}$, indeed: infinite chattering is impossible since we cannot follow $\gtrsim$ forever, and interaction cannot fail after following a finite portion of $\gtrsim$ since those finite portions $\gtrsim_i$ are in $V_{\mathbf{A}}$. Hence $\mathfrak{e} \in \mathbf{A}^\perp$. But $\mathfrak{d} \not\perp \mathfrak{e}$, because of infinite chattering following $\gtrsim$. Contradiction.

If $\mathfrak{d}$ has a finite number of maximal slices $\mathfrak{c}_1, \dots, \mathfrak{c}_k$ then for every $i \leq k$ there exist an ∞-path $\mathfrak{s}_i$ that visit all the positive proper actions of $\mathfrak{c}_i$. Indeed, any (either infinite or positive-ended) sequence $\mathfrak{s}$ of proper actions in a slice $\mathfrak{c} \sqsubseteq \mathfrak{d}$, without repetition, such that polarities alternate and the views of prefixes of $\mathfrak{s}$ are views of $\mathfrak{c}$, is an ∞-path:

- (Linearity) is ensured by the fact that we are in only one slice,
- (O-visibility) is satisfied since positive actions of $\mathfrak{d}$, thus also of $\mathfrak{c}$, are justified by the immediate previous negative action (a condition true in $|\mathbf{A}|$, thus also satisfied in $\mathfrak{d}$ because all its views are views of designs in $|\mathbf{A}|$)
- (P-visibility) is natively satisfied by the fact that $\mathfrak{s}$ is a promenade in the tree representing a design.

For example, $\mathfrak{s}$ can travel in the slice $\mathfrak{c}$ as a breadth-first search on couples of nodes (κ^-, κ^+) such that κ^+ is just above κ^- in the tree, and κ^+ is proper. Then 2 cases:

- Either for all i, there exists $n_i \in \mathbb{N}$ and $\mathfrak{d}_i \in \mathbf{A}_{n_i}$ such that $\mathfrak{s}_i$ is an ∞-path of $\mathfrak{d}_i$. Without loss of generality we can even suppose that $\mathfrak{c}_i \sqsubseteq \mathfrak{d}_i$: if it is not the case, replace some positive subdesigns (possibly Ω) of $\mathfrak{d}_i$ by $\maltese$ until you obtain $\mathfrak{d}'_i$ such that $\mathfrak{c}_i \sqsubseteq \mathfrak{d}'_i$, and note that indeed $\mathfrak{d}'_i \in \mathbf{A}_{n_i}$ since $\mathfrak{d}_i \preceq \mathfrak{d}'_i$. Let $N = \max_{1\leq i\leq k}(n_i)$. Since $\mathfrak{d} \notin \mathbf{A}$, thus in particular $\mathfrak{d} \notin \mathbf{A}_N$, there exists $\mathfrak{e} \in \mathbf{A}_N^\perp$ such that $\mathfrak{d} \not\perp \mathfrak{e}$. The reason of divergence cannot be infinite chattering, otherwise there would exist an infinite ∞-path t in $\mathfrak{d}$ such that $\tilde{t}$ is in $\mathfrak{e}$, and t is necessarily in a single slice of $\mathfrak{d}$ (say $\mathfrak{c}_i$) to ensure its linearity; but in this case we would also have $\mathfrak{d}_i \not\perp \mathfrak{e}$ where $\mathfrak{d}_i \in \mathbf{A}_N$, impossible. Similarly, for all (finite) path $\mathfrak{s}$ of $\mathfrak{d}$, there exists i such that $\mathfrak{s}$ is a path of $\mathfrak{c}_i$ thus of $\mathfrak{d}_i \in \mathbf{A}_N$; this ensures that interaction between $\mathfrak{d}$ and $\mathfrak{e}$ cannot diverge after a finite number of steps either, leading to a contradiction.
- Or there is an i such that the (necessarily infinite) ∞-path $\mathfrak{s}_i$ is in no design of $\mathbf{A}$. In this case, let $\mathfrak{e} = {}^{\maltese\overset{\sim}{\mathfrak{s}_i}\dashv^c}$ (where $\overset{\sim}{\mathfrak{s}_i}$ is a view since the $\mathbf{A}_n$ are simple), and with a similar argument as previously we have $\mathfrak{e} \in \mathbf{A}^\perp$ but $\mathfrak{d} \not\perp \mathfrak{e}$ by infinite chattering, contradiction. ◀

Advice Automatic Structures and Uniformly Automatic Classes

Faried Abu Zaid[1], Erich Grädel[2], and Frederic Reinhardt[3]

1 **Department of Computer Science and Automation, TU Ilmenau, Ilmenau, Germany**
Faried.Abu-Zaid@tu-ilmenau.de

2 **Mathematical Foundations of Computer Science, RWTH Aachen University, Aachen, Germany**
graedel@logic.rwth-aachen.de

3 **Mathematical Foundations of Computer Science, RWTH Aachen University, Aachen, Germany**
reinhardt@logic.rwth-aachen.de

Abstract

We study structures that are automatic with advice. These are structures that admit a presentation by finite automata (over finite or infinite words or trees) with access to an additional input, called an advice. Over finite words, a standard example of a structure that is automatic with advice, but not automatic in the classical sense, is the additive group of rational numbers $(\mathbb{Q}, +)$.

By using a set of advices rather than a single advice, this leads to the new concept of a parameterised automatic presentation as a means to uniformly represent a whole class of structures. The decidability of the first-order theory of such a uniformly automatic class reduces to the decidability of the monadic second-order theory of the set of advices that are used in the presentation. Such decidability results also hold for extensions of first-order logic by regularity preserving quantifiers, such as cardinality quantifiers and Ramsey quantifiers.

To investigate the power of this concept, we present examples of structures and classes of structures that are automatic with advice but not without advice, and we prove classification theorems for the structures with an advice automatic presentation for several algebraic domains. In particular, we prove that the class of all torsion-free Abelian groups of rank one is uniformly ω-automatic and that there is a uniform ω-tree-automatic presentation of the class of all Abelian groups up to elementary equivalence and of the class of all countable divisible Abelian groups. On the other hand we show that every uniformly ω-automatic class of Abelian groups must have bounded rank.

While for certain domains, such as trees and Abelian groups, it turns out that automatic presentations with advice are capable of presenting significantly more complex structures than ordinary automatic presentations, there are other domains, such as Boolean algebras, where this is provably not the case. Further, advice seems to not be of much help for representing some particularly relevant examples of structures with decidable theories, most notably the field of reals.

Finally we study closure properties for several kinds of uniformly automatic classes, and decision problems concerning the number of non-isomorphic models in uniformly automatic classes with the unique representation property.

1998 ACM Subject Classification F.4.1 [Mathematical Logic] Model Theory

Keywords and phrases automatic structures, algorithmic model theory, abelian groups, torsion-free abelian groups, first-order logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.35

1 Introduction

Automatic structures are structures that allow finite presentations by automata. Roughly speaking, a structure is called automatic if its domain can be represented as a regular set in such a way that its relations become recognisable by synchronous multi-tape automata.

The history of automatic structures can be traced back to the early days of automata theory, for instance to the automata theoretic decision procedures by Büchi and Rabin for Presburger arithmetic and other theories. A more systematic investigation has been started by Khoussainov and Nerode [14], who also coined the term automatic structures. In [5] the concept was lifted from finite words to automata that read trees as well as their infinite counterparts. For a more elaborate introduction to the topic we refer the reader to [3, 20].

An important research objective in the field of automatic structures is to determine which structures admit automatic presentations and to characterise all automatic models inside certain classes of structures. For instance, a long standing open problem had been whether the additive group of the rational numbers is automatic, until Tsankov [23] gave a negative answer to the question. It has been noted, however, that $(\mathbb{Q}, +)$ is "almost" automatic in the sense that there is a presentation in which addition is automatic but the domain is not a regular set [19]. Kruckman et al. remarked in [16] that the domain is also recognisable by an automaton, provided that it has access to a specific infinite advice string. Moreover this advice string itself has a decidable monadic second-order theory, which is sufficient to give an automata-based decision procedure for the first-order theory of $(\mathbb{Q}, +)$.

This motivates our study of advice automatic structures. A structure is advice automatic if it has an automatic presentation in the same way as $(\mathbb{Q}, +)$ does: it can be presented by automata that have access to some fixed advice. This setting has appeared occasionally in the literature [7, 13] but to the authors knowledge no systematic investigation has been done so far. Advice automatic structures are interesting, because they generalise the domain of infinite structures that admit automata-based finite presentations while, as we shall prove, preserving the good algorithmic and model-theoretic properties of automatic structures, in particular the decidability of their first-order theories. But there is a further very interesting twist: Automata with advice permit us to lift the notion of an automatic presentation from single structures to classes of structures that can be represented by a single presentation, but with a set of different advices. This will lead us to the concept of uniformly automatic classes of structures.

We shall in fact introduce several variants of this concept. Of course, not all advice sets give us classes of structures with a decidable theory since one can easily encode undecidable problems inside the set of advices, or even in a single advice. But any class of structures that admits an automatic presentation with an advice set that has a decidable monadic second-order theory does indeed have an automata-based decision procedure for its first-order theory, and even for the extension of first-order logic by different variants of cardinality quantifiers and by Ramsey quantifiers. These results show that automatic presentations with advice provide relevant generalisations of the concept of automata-based representations of infinite structures, and that the algorithmic properties, which make automatic structures suitable for applications, survive under these generalisations.

We then investigate the power of this concept. We identify classes of structures, such as trees and Abelian groups, where automatic presentations with advice are capable of presenting significantly more complex structures than ordinary automatic presentations. Among other results we provide a uniformly ω-automatic presentation of the torsion-free Abelian groups of rank one and a uniformly ω-tree automatic presentation for the class of all countable divisible Abelian groups and the class of all Abelian groups up to elementary

equivalence. We also prove limitations of this concept by identifying classes, such as the class of all Boolean algebras, where we do not gain anything essential from the access to an advice. We further show that every uniformly ω-automatic class of countable Abelian groups must have bounded rank and extend known non-automaticity results to the case of advice automatic structures. In particular, it turns out that an advice does not help for representing some particularly relevant examples of structures with decidable theories, most notably the field of reals.

Further we investigate closure properties for uniformly automatic classes. We show that whenever a class of structures is uniformly tree- or ω-tree-automatic then this is also true for the closure under direct products and the closure under disjoint unions, a property that is not shared by the uniformly ω-automatic classes.

Finally, we study decidability issues for counting the number of non-isomorphic models inside uniformly automatic classes that have the unique representation property, i.e. where distinct advices always give non-isomorphic structures. While such counting problems often are decidable, the unique representation property itself turns out to be undecidable even in the simplest conceivable cases of regularly automatic classes.

2 Automatic Presentations with Advice

For two words $v, w \in \Sigma^*$ the **convolution** $v \otimes w$ is a word over the alphabet $(\Sigma \uplus \{\Box\})^2$ of length $\max(|v|, |w|)$ with

$$(v \otimes w)(i) = \begin{cases} (v(i), w(i)) & \text{if } i < \min(|v|, |w|) \\ (\Box, w(i)) & \text{if } |v| \leq i < |w| \\ (v(i), \Box) & \text{if } |w| \leq i < |v|. \end{cases}$$

The convolution of two ω-words is defined analogously with the difference that a padding symbol is not needed. The convolution of trees follows the same idea. For two Σ-labelled trees s, t the convolution $s \otimes t$ is the $(\Sigma \uplus \{\Box\})^2$-labelled tree with $\mathrm{dom}_{s\otimes t} = \mathrm{dom}_s \cup \mathrm{dom}_t$ and the labelling

$$(s \otimes t)(w) = \begin{cases} (s(w), t(w)) & \text{if } w \in \mathrm{dom}_s \cap \mathrm{dom}_t \\ (\Box, t(w)) & \text{if } w \in \mathrm{dom}_t \setminus \mathrm{dom}_s \\ (s(w), \Box) & \text{if } w \in \mathrm{dom}_s \setminus \mathrm{dom}_t. \end{cases}$$

Instead of $w_1 \otimes w_2 \otimes \cdots \otimes w_n$ we will often write $\langle w_1, w_2, \ldots, w_n \rangle$, and for a language L we let $L^{\otimes n} := \{\langle w_1, \ldots, w_n \rangle \mid w_1, \ldots, w_n \in L\}$. For $I \subseteq \mathbb{N}$ let wI denote the subword of w that consists of the letters at positions in I. We denote the **prefix-relation** on words by $<_p$, the **length-lexicographical relation** by $<_{llex}$, and for $m \in \mathbb{N}$ the **m-equal-ends** relation $\sim_e^m$ on infinite words is given by $v \sim_e^m w$ if, and only if, $v[m, \infty) = w[m, \infty)$. The **equal-ends** relation $\sim_e$ is the union of the $\sim_e^m$, i.e. $v \sim_e w$ if and only if $v \sim_e^m w$ for some $m \in \mathbb{N}$.

We assume familiarity with the classical models of finite automata and extend their semantics to define languages that are *regular with advice*.

▶ **Definition 1.** A **parameterised Muller automaton** is a Muller automaton $\mathcal{A}$ over the alphabet $\Sigma \times \Gamma$. For $\alpha \in \Gamma^\omega$. **the language that $\mathcal{A}$ recognises with advice** α is $L(\mathcal{A}[\alpha]) := \{\beta \in \Sigma^\omega \mid \beta \otimes \alpha \in L(\mathcal{A})\}$. In this case we also say L is recognised by $\mathcal{A}[\alpha]$. A language L is called **ω-regular with advice** α if there is a parameterised Muller automaton $\mathcal{A}$ with $L = L(\mathcal{A}[\alpha])$.

Parameterised automata on finite words and finite or infinite trees are defined analogously.

An automaton $\mathcal{A}$ **recognises a relation** (possibly with advice) R if it recognises the language $\{a_1 \otimes \ldots \otimes a_k \mid (a_1, \ldots, a_k) \in R\}$ of all convolutions of tuples in R. Given a relation $R \subseteq A^s \times A^t$ and tuples $\overline{a} \in A^s$, $\overline{b} \in A^t$, we will frequently also use the notation $\overline{a}R$ and $R\overline{b}$ to denote the projections $\{\overline{b} \in A^t \mid (\overline{a}, \overline{b}) \in R\}$ and $\{\overline{a} \in A^s \mid (\overline{a}, \overline{b}) \in R\}$, respectively. We are now ready to introduce the notion of an automatic presentation with advice. It is a general concept but we state it for automata on infinite words.

▶ **Definition 2.** Let τ be a finite relational signature. An **ω-automatic presentation with advice** α is a tuple $\mathfrak{d} = (\mathcal{A}, \mathcal{A}_\approx, (\mathcal{A}_R)_{R\in\tau})$ of parameterised Muller automata such that:

- $A_\alpha := L(\mathcal{A}[\alpha])$ presents the universe of a structure.
- For $R \in \tau$ of arity r, $\mathcal{A}_R[\alpha]$ recognises an r-ary relation R_α on A_α.
- $\mathcal{A}_\approx[\alpha]$ recognises a binary congruence relation $\approx_\alpha$ on the structure $(A_\alpha, (R_\alpha)_{R\in\tau})$.

The **induced structure of $\mathfrak{d}$ and α** is $\mathcal{S}_\approx(\mathfrak{d}[\alpha]) := (A_\alpha, \approx_\alpha, (R_\alpha)_{R\in\tau})$ and we say that $\mathfrak{d}[\alpha]$ **presents the structure** $\mathcal{S}(\mathfrak{d}[\alpha]) := \mathcal{S}_\approx(\mathfrak{d}[\alpha])/\approx_\alpha$. In the case that $\approx_\alpha$ is just the identity we say that **the presentation is injective** and omit $\mathcal{A}_\approx$ in our notation.

We say that a τ-structure $\mathfrak{A}$ is **ω-automatic with advice** if there is a parameterised ω-automatic presentation $\mathfrak{d}$ with $\mathfrak{A} \cong \mathcal{S}(\mathfrak{d}[\alpha])$ for some parameter α. In our applications we will often assume that we have also fixed a witnessing isomorphism $\pi : \mathcal{S}(\mathfrak{d}[\alpha]) \to \mathfrak{A}$. Note that such an isomorphism π extends in a natural way to a strong homomorphism $\pi_\approx : \mathcal{S}_\approx(\mathfrak{d}[\alpha]) \to \mathfrak{A}$. Further we extend $\pi_\approx$ to convolutions of words in $L(\mathcal{A}[\alpha])$ in the obvious way.

By changing the automata model we obtain analogous notions of structures that are, for instance, word-automatic with advice or tree-automatic with advice.

3 Uniformly Automatic Classes

Although to our knowledge the concept of a uniformly automatic class has not been explicitly studied in literature there are of course several examples where the underlying idea has been very successfully applied in various areas of computer science. This includes the following insights:

- The class of all *countable linear orders* is regularly ω-tree-automatic.
- For any fixed $d \in \mathbb{N}$, the class of finite graphs of *tree width* at most d and the class of finite graphs of *clique width* at most d are regularly tree-automatic.
- For any fixed $d \in \mathbb{N}$, the class of finite graphs of *path width* at most d and the class of finite graphs of *linear clique width* at most d are regularly automatic.

To make this precise and more general, we introduce the following definitions. As above, we state them in terms of automata on infinite words. The corresponding variants based on automata on finite words, or on finite or infinite trees are completely analogous.

▶ **Definition 3.** A class of τ-structures $\mathcal{C}$ is **uniformly ω-automatic** if there is a parameterised ω-automatic presentation $\mathfrak{c}$ and a set of parameters P, so that $\mathcal{S}(\mathfrak{c}[P]) \coloneqq \{\mathcal{S}(\mathfrak{c}[\alpha]) \mid \alpha \in P\}$ is equal to $\mathcal{C}$ up to isomorphism. If P has a decidable MSO-theory we say that $\mathcal{C}$ is **strongly ω-automatic**. If P is even regular then we say that $\mathcal{C}$ is **regularly ω-automatic**. In this case we call a tuple $(\mathcal{A}_p, \mathfrak{c})$ with $L(\mathcal{A}_p) = P$ a **regularly ω-automatic presentation** of $\mathcal{C}$.

What makes automatic structures so interesting for applications in computer science is that there is an effective decision procedure for the FO-theory of every automatic structure. We outline how to get an effective decision procedure for the FO-theory of a strongly

automatic class. The decision procedure works by recursively building the union, complement or projection automaton from automata that recognise the relations defined by subformulae. Since advice automata are basically ordinary synchronous multi-tape automata with a designated advice tape, advice regular relations are also effectively closed under union, complement and projection and thus the following theorem holds.

▶ **Theorem 4.** *There is an algorithm which, given a parameterised ω-automatic presentation $\mathfrak{d}$ and a* FO*-formula $\varphi(\overline{x})$ over the signature of $\mathfrak{d}$, constructs an automaton $\mathcal{A}_\varphi$ with $L(\mathcal{A}_\varphi) = \{\overline{a} \otimes \alpha \mid \mathcal{S}_\approx(\mathfrak{d}[\alpha]) \models \varphi(\overline{a})\}$.*

Given a sentence φ the algorithm constructs an automaton with $L(\mathcal{A}_\varphi) = \{\alpha \mid \mathcal{S}_\approx(\mathfrak{d}[\alpha]) \models \varphi\}$. Deciding whether φ is in the FO-theory of a class $\mathcal{S}(\mathfrak{d}[P])$ thus reduces to deciding the inclusion problem $P \subseteq L(\mathcal{A}_\varphi)$. The well-known correspondence theorems between MSO-definable languages and regular languages imply that there is a MSO-sentence ψ with $\alpha \models \psi$ if, and only if, $\alpha \in L(\mathcal{A}_\varphi)$ and thus the inclusion problem reduces to checking whether ψ holds in every $\alpha \in P$, which proves claims 2 and 3 of the following corollary. Claims 1 and 4 follow from Theorem 4 analogously to the case of automatic structures. We refer to [4] for an introduction to FO-interpretations in the context of automatic structures.

▶ **Corollary 5.**

1. *The class of ω-automatic structures with advice α is effectively closed under* FO*-interpretations.*
2. *The* FO*-theory of a structure that is ω-automatic with advice α is decidable if the* MSO*-theory of α is decidable.*
3. *The* FO*-theory of a strongly ω-automatic class is decidable.*
4. *If $\mathcal{C}$ is* FO*-interpretable in a uniformly ω-automatic class $\mathcal{D}$ then $\mathcal{C}$ is also uniformly ω-automatic.*

The analogous automatic, tree- and ω-tree-automatic versions of these statements hold true as well.

Analogous versions of Theorem 4 further hold for extensions of FO by regularity preserving quantifiers, i.e. evaluation of a formula with regularity preserving quantifiers in an automatic structure yield effectively a regular relation again. We will later make use of the cardinality quantifiers $\exists^\infty / \exists^{>\aleph_0} / \exists^{(k,m)}$, meaning "there exists infinitely/uncountably/k mod m/ many", which are regularity preserving for automatic and ω-automatic structures [12]. FOC denotes the extension of FO by $\exists^\infty$,$\exists^{>\aleph_0}$,$\exists^{(k,m)}$.

It is known that every countable ω-automatic structure is automatic [12]. We can generalise this result to uniformly ω-automatic classes of countable structures in the following sense: We say that an ω-automatic presentation is a **presentation over finite words** if the elements of the domain(s) of the structure(s) are encoded in a subset of $\Sigma^*\{\square\}^\omega$. When a finite words presentation is given we will for brevity often write w for $w\square^\omega$.

▶ **Theorem 6.** *A class $\mathcal{C}$ of countable structures has a parameterised ω-automatic presentation $\mathfrak{c}$ with parameter set P, if, and only if, it has an injective parameterised ω-automatic presentation over finite words $\mathfrak{c}'$ with the same parameter set P. Moreover $\mathfrak{c}'$ can be effectively constructed from $\mathfrak{c}$.*

Another example for a quantifier that is regularity preserving for automatic structures is the Ramsey quantifier. For any $k \geq 1$, the *k-Ramsey quantifier* $\exists^{\text{k-ram}}$ is defined by $\mathfrak{A} \models \exists^{\text{k-ram}} \overline{x} \varphi(\overline{x}, \overline{c})$ if, and only if, there is an infinite $X \subseteq A$ so that $\mathfrak{A} \models \varphi(a_1, \ldots, a_k, \overline{c})$ for all pairwise different $a_1, \ldots, a_k \in X$.

The Ramsey quantifier is not regularity preserving anymore for ω-automatic structures [17]. As the following lemma implies it is however regularity preserving for countable ω-automatic structures with advice. We will also make use of the following lemma in the proof of Theorem 17.

▶ **Lemma 7.** *Let $\mathfrak{d}$ be an ω-automatic presentation over finite words with advice α, and $P = \{\overline{p} \in A_\alpha^l \mid \mathcal{S}_\approx(\mathfrak{d}[\alpha]) \models \exists^{\text{k-ram}} \overline{x} R_\alpha \overline{x}\overline{p}\}$, where R_α is a $(k+l)$-ary relation of $\mathcal{S}_\approx(\mathfrak{d}[\alpha])$. Then there is a subset $A' \subseteq A_\alpha \otimes P$ that is ω-regular with advice α so that for all $\overline{p} \in P$ the sets $A'\overline{p} := \{a \in A_\alpha \mid a \otimes \overline{p} \in A'\}$ are infinite with $\mathcal{S}_\approx(\mathfrak{d}[\alpha]) \models R_\alpha(a_1, \ldots, a_k, \overline{p})$ for all pairwise different $a_1, \ldots, a_k \in A'\overline{p}$.*

A parameterised Muller automaton $\mathcal{A}'$ with $L(\mathcal{A}'[\alpha]) = A'$ can be effectively constructed.

Proof. Let Σ be the alphabet of the presentation $\mathfrak{d}$. Consider ω-words of the form $s \otimes t$ with $s = s_0 s_1 \ldots$ and $t = t_0 t_1 \ldots$ such that $|s_i| = |t_i|$ and $s_i \in \Sigma^* \{\dot{a} \mid a \in \Sigma\}, t_i \in \Sigma^*$. Say that a word $x \in \Sigma^*$ is *on* $s \otimes t$, if there is an i so that $x = s_0 \ldots s_i t_{i+1}$ (ignoring dots on letters). Let $On(s \otimes t)$ be the set of words that are on $s \otimes t$. It is not hard to construct a parameterised Muller automaton $\mathcal{A}$, so that $\mathcal{A}[\alpha]$ recognises exactly those ω-words of the form $s \otimes t \otimes \overline{p}$ with $\mathcal{S}_\approx(\mathfrak{d}[\alpha]) \models R_\alpha(x_1, \ldots, x_k, \overline{p})$ for all pairwise different $x_1, \ldots, x_k \in On(s \otimes t)$. Applying the uniformization theorem for ω-regular relations [6] to the $(2+l+1)$-ary relation recognised by $\mathcal{A}$, we get a Muller automaton $\mathcal{U}$ so that for every $\overline{p} \in P$ there is *at most* one $s \otimes t$ with $s \otimes t \otimes \overline{p} \otimes \alpha \in L(\mathcal{U}) \Leftrightarrow s \otimes t \otimes \overline{p} \in L(\mathcal{U}[\alpha])$. From $\mathcal{U}$ we can easily construct another parameterised Muller automaton $\mathcal{A}'$ which on input $\overline{p}$ guesses a $s \otimes t$ with $s \otimes t \otimes \overline{p} \in L(\mathcal{U}[\alpha])$ and uses it to recognise $On(s \otimes t) \otimes \overline{p}$, i.e. $L(\mathcal{A}'[\alpha]) = \{w \otimes \overline{p} \mid w \in On(s \otimes t)\}$. It remains to show that for each $\overline{p} \in P$ there is at least one $s \otimes t$ with $s \otimes t \otimes \overline{p} \in L(\mathcal{A})$. Let $\overline{p} \in P$ and $X \subseteq \Sigma^*$ be an infinite set with $\mathcal{S}_\approx(\mathfrak{d}[\alpha]) \models R_\alpha(x_1, \ldots, x_k, \overline{p})$ for all pairwise distinct $x_1, \ldots, x_k \in X$. Consider the subtree of $(\Sigma^*, \leq_p)$ that is generated by the prefix-closure of X. According to König's Lemma there is an infinite path $\gamma \in \Sigma^\omega$ in this tree so that from every node on the path a node in X is reachable. We define inductively words $s_i, t_i \in \Sigma^*$, so that the following invariants hold: 1. $|s_i| = |t_i|$ for all $i \in \mathbb{N}$, 2. $s_0 \ldots s_i$ is a prefix of γ for all $i \in \mathbb{N}$ and 3. $s_0 \ldots s_i t_{i+1} \in X$ for all $i \in \mathbb{N}$. Define $s_0 := \varepsilon, t_0 := \varepsilon$, t_{i+1} as a shortest path from $s_0 \ldots s_i$ to a node in X and s_{i+1} as the path of length $|t_{i+1}|$ so that $s_0 \ldots s_{i+1}$ remains a prefix of γ for all $i \in \mathbb{N}$. ◀

4 Examples and Classifications for Uniformly Automatic Classes

We shall now provide examples of structures that admit automatic presentations with advice, and of uniformly automatic classes. For every concept of structures, and classes of structures, that admit a certain type of finite presentation, it is of course relevant to understand which structures and classes actually fall under this concept. We study this question here for infinite trees, for Abelian groups, for Boolean algebras and some further algebraic domains.

4.1 Infinite Trees

Consider the infinite $|\Sigma|$-ary tree $\mathfrak{S}_\Sigma := (\Sigma^*, \leq_p, (S_a)_{a \in \Sigma}, \text{el})$ with successor relations $S_a := \{(w, wa) : w \in \Sigma^*\}$ and equal-level relation $\text{el} := \{(w, v) \in \Sigma^* \times \Sigma^* : |w| = |v|\}$. It is well known that $\mathfrak{S}_\Sigma$ has an automatic presentation [5]. In the following we want to consider two ways to generalise the tree $\mathfrak{S}_\Sigma$ to obtain two different types of uniformly automatic classes of infinite trees with equal-level relation el and ancestor relation $\leq_p$. In the first generalisation we only consider trees with bounded node degree, but relax the condition that the subtree at each node has the same isomorphism type, which is the property that characterises the

trees $\mathfrak{S}_\Sigma$. Instead we allow that the trees have on each level i at most $|Q|$ many different isomorphism types of subtrees for a constant $|Q|$. Trees of this kind can also be characterised as the unwindings of advice automata, where each path follows at each step i a transition from a new transition relation $\Delta(i) \subseteq Q \times \Sigma \times Q$.

▶ **Example 8.** Let Q, Σ be finite sets, $q_0 \in Q$ and $\Delta \subseteq Q \times \Sigma \times Q \times \mathbb{N}$.

The tree $\mathfrak{S}_{\Sigma,\Delta} := (\tilde{\Delta}, \leq_p, (\tilde{\Delta}_a)_{a\in\Sigma}, \mathrm{el})$ that consists of all Σ-labelled finite paths $\tilde{\Delta} := \{(q_0, a_0, q_1)(q_1, a_1, q_2)(q_2, a_2, q_3)...(q_n, a_n, q_{n+1}) \in (Q \times \Sigma \times Q)^* : n \in \mathbb{N}, (q_i, a_i, q_{i+1}, i) \in \Delta\}$ with successor relations $\tilde{\Delta}_a := \{(w, v) \in \tilde{\Delta} \times \tilde{\Delta} : v = w(q, a, p) \wedge (q, a, p, |w|) \in \Delta\}$ has an ω-automatic presentation with advice.

Furthermore for any fixed Q, Σ the class $\mathcal{T}_{Q,\Sigma} := \{\mathfrak{S}_{\Sigma,\Delta} : \Delta \subseteq Q \times \Sigma \times Q \times \mathbb{N}\}$ is regularly ω-automatic. To see this, let $\Gamma := \mathcal{P}(Q \times \Sigma \times Q)$ be the advice alphabet. It is easy to construct a uniform ω-automatic presentation that represents $\mathfrak{S}_{\Sigma,\Delta}$ with the advice $\Delta(0)\Delta(1)\ldots \in \Gamma^\omega$, where $\Delta(i) := \{(q, a, p) : (q, a, p, i) \in \Delta\}$ for all $i \in \mathbb{N}$. Moreover the set of parameters Γ^ω is an ω-regular set.

Next we consider the class of trees with the property that each node has finite degree and all nodes of the same depth have the same number of a-successors for each $a \in \Sigma$.

▶ **Example 9.** $\mathcal{C}_\Sigma := \{(T, \leq_p, (S_a)_{a\in\Sigma}, \mathrm{el}) : \forall a \in \Sigma \forall (t, t') \in \mathrm{el} : |tS_a| < \infty \wedge |tS_a| = |t'S_a|\}$, where tS_a denotes the set of a-successors of t, is uniformly ω-automatic.

Choose $\Gamma := \Sigma \cup \{\#\}$ as the advice alphabet. Then the tree $(T, \leq_p, (S_a)_{a\in\Sigma}, \mathrm{el}) \in \mathcal{C}_\Sigma$ can be represented with an advice of the form $\alpha := \alpha_0\#\alpha_1\#\ldots$ where $\alpha_i \in \Sigma^*$ such that for all $a \in \Sigma$ $|\alpha_i|_a = |tS_a|$ for any $t \in T$ of depth i. Code the domain of the tree by $D_T := \{w_0\#w_1\#\ldots\#w_n : n \in \mathbb{N}, \text{ for all } i \leq n : w_i \in 0^*10^* \wedge |w_i| = |\alpha_i|\}$ which is obviously regular with advice α. $\leq_p$ and el are just the regular prefix-relation and equal-length relation on words and $S_a := \{(x, x\#0^i10^k) \in D_T \times D_T : x = w_0\#w_1\#...\#w_{n-1} \wedge \alpha_n(i) = a\}$ is also regular with advice α.

4.2 Abelian Groups

We recall some standard notions and facts of Abelian group theory. The *order* of an element a in an Abelian group is the smallest positive integer n with $n \cdot a = 0$, or ∞ if no such n exists. A group is *torsion-free*, if the neutral element is the only element of finite order in the group. A group is *periodic*, if all of its elements have finite order. The *rank* of an Abelian group is the cardinality of a maximal subset S of the group that is linearly independent over $\mathbb{Z}$, i.e. such that for any nonempty finite subset $F \subseteq S$ the equation $\sum_{a\in F} z_a \cdot a = 0$ in the variables $(z_a)_{a\in F}$ has over $\mathbb{Z}$ only the trivial solution $z_a = 0$ for all $a \in F$.

The torsion-free Abelian groups of rank n coincide up to isomorphism with the subgroups of $(\mathbb{Q}^n, +)$ [8], so that it is sufficient to consider a classification of those. A complete classification of the subgroups of $(\mathbb{Q}, +)$ has long been known [2]. The classification problem of the torsion-free Abelian groups of rank n for $n \geq 2$ on the other side seems to be much more intricate and is an active research area of infinite Abelian group theory [22]. We recall here Baer's classification of the subgroups of $(\mathbb{Q}, +)$. Let $\mathbb{P}$ be the set of prime numbers. Every sequence $c := (c_p)_{p\in\mathbb{P}}$ with $c_p \in \mathbb{N} \cup \{\infty\}$ for all $p \in \mathbb{P}$ corresponds to the subgroup $(\mathbb{Q}_c, +) := \left(\left\{\frac{z}{p_1^{d_1}\cdots p_k^{d_k}} \mid z \in \mathbb{Z}, p_i \in \mathbb{P}, d_i \in \mathbb{N}, d_i \leq c_{p_i}\right\}, +\right)$ of $(\mathbb{Q}, +)$ and every subgroup of $(\mathbb{Q}, +)$ is isomorphic to a group of the form $(\mathbb{Q}_c, +)$ for some c.

It has already been noted in [16, 19] that addition of rational numbers is advice automaton recognizable, if rationals are encoded as digit sequences $(d_i)_i$ of their *factorial base*

representation. Every rational number $r \in [0,1)$ can be written as $r = \sum_{i=2}^{n} \frac{d_i}{i!}$ with $d_i < i$. The following lemma generalises the factorial base representation, so that all and only the elements of a subgroup $(\mathbb{Q}_c, +)$ have a representation in a generalised factorial base which depends on c. For this sake we substitute for a given sequence of natural numbers $n = (n_i)_{i\in\mathbb{N}}$ the factorial $i!$ by a generalised factorial $n_{\underline{i!}} \coloneqq n_{i-1}n_{i-2}\dots n_0$. We define the index notation $\underline{z!}$ for any $z \in \mathbb{Z}$ via $n_{\underline{z!}} \coloneqq (\prod_{i<|z|} n_i)^{\operatorname{sgn}(z)}$, where $\operatorname{sgn}(z)$ is the sign of z. Further let $h_p(n)$ be the exponent of p in the prime factorisation of n where $n \in \mathbb{N}, p \in \mathbb{P}$. The precise conditions under which all and only the elements of $\mathbb{Q}_c$ have a unique presentation as a digit sequence in a suitable generalised factorial base are given in the next lemma.

▶ **Lemma 10.** *Let $(n_i)_{i\in\mathbb{N}}$ be a sequence of natural numbers with $n_i \geq 2$ for all $i \in \mathbb{N}$ and let $c \coloneqq (c_p)_{p\in\mathbb{P}}$ with $c_p \coloneqq \sum_{i=0}^{\infty} h_p(n_i)$ for all $p \in \mathbb{P}$ (set $\sum_{i=0}^{\infty} h_p(n_i) = \infty$ if the sum does not converge). Then for every $r \in \mathbb{Q}_c$ with $r \geq 0$ there is a unique sequence $(d_z)_{z=-l}^{k}$ with*

1. *$0 \leq d_i < n_i$ for $i = 0,\dots,k$ and $0 \leq d_{-i} < n_{i-1}$ for $i = 1,\dots,l$*
2. *$d_k \neq 0$ or $k = 0, d_k = 0$*
   $d_{-l} \neq 0$ or $l = 1, d_{-l} = 0$
3. *$r = \sum_{z=-l}^{k} d_z n_{\underline{z!}}$*

Proof. Let $r \in \mathbb{Q}_c$ with $r \geq 0$. Decompose r into its fractional part and its integer part $r = \frac{a}{b} + m$ with $m, a, b \in \mathbb{N}$,$0 \leq \frac{a}{b} < 1$, and a, b coprime.

First we show how to get $d_{-l},\dots,d_{-1}$ with $\frac{a}{b} = \sum_{z=-l}^{-1} d_z n_{\underline{z!}}$. Since $c_p \geq h_p(b)$ for any prime factor p of b there is an $l(p)$ with $h_p(n_0\dots n_{l(p)}) = h_p(n_0) + \dots + h_p(n_{l(p)}) \geq h_p(b)$. Thus for $l - 1 = \max\{l(p) \mid p \in \mathbb{P}, p|b\}$ it holds that $b|n_0\dots n_{l-1}$. Consequently there is a q with $\frac{a}{b} = \frac{qa}{n_0\dots n_{l-1}}$. If $qa < n_{l-1}$ let $d_{-i} \coloneqq 0$ for $i < l$ and $d_{-l} \coloneqq qa$ and be done. Otherwise $qa \geq n_{l-1}$ and $l - 1 > 0$, then $qa = n_{l-1}q' + d_{-l}$ for some $q', d_{-l} \in \mathbb{N}$ with $d_{-l} < n_{l-1}$. Then $\frac{a}{b} = \frac{d_{-l}}{n_0\dots n_{l-1}} + \frac{q}{n_0\dots n_{l-2}}$ and we can continue the decomposition recursively to obtain $d_{-1},\dots,d_{-(l-1)}$ with $\frac{q}{n_0\dots n_{l-2}} = \sum_{z=-(l-1)}^{-1} d_z n_{\underline{z!}}$.

Similarly we show how to get $d_0,\dots,d_k$ with $m = \sum_{z=0}^{k} d_z n_{\underline{z!}}$. Choose the smallest k so that $m < n_0\dots n_k$. If $k = 0$ let $d_0 \coloneqq m$ and be done. Otherwise $n_0\dots n_{k-1} \leq m < n_0\dots n_k$ and thus $m = d_k n_0\dots n_{k-1} + q$ for some $d_k < n_k$ and $q \in \mathbb{N}$ and we can continue the decomposition recursively with $q < n_0\dots n_{k-1}$ to obtain $d_0,\dots,d_{k-1}$ with $q = \sum_{z=0}^{k-1} d_z n_{\underline{z!}}$.

It remains to prove that the representations are unique. First note that $0 \leq \sum_{z=0}^{k} d_z n_{\underline{z!}} \leq \sum_{z=0}^{k}(n_z - 1)n_{\underline{z!}} = \sum_{z=0}^{k} n_{z+1!} - n_{\underline{z!}} = n_{\underline{k+1!}} - 1$. The mapping that maps each digit sequence $(d_z)_{z=0}^{k}$ to $\sum_{z=0}^{k} d_z n_{\underline{z!}}$ is, as was shown above, surjective and since there are only $n_{\underline{k+1!}}$ many such digit sequences it must also be injective.

Now suppose $(d_z)_{z=-l}^{-1}$ and $(d'_z)_{z=-l'}^{-1}$ would be two different representations of $0 \leq \frac{a}{b} < 1$. Let $l \geq r \geq 1$ be the smallest number with $d_{-r} \neq d'_{-r}$. Then $d_{-r} + \frac{d_{-r-1}}{n_r} + \dots + \frac{d_{-l}}{n_r\dots n_{l-1}} = d'_{-r} + \frac{d'_{-r-1}}{n_r} + \dots + \frac{d'_{-l'}}{n_r\dots n_{l'-1}}$. Since $\frac{d_{-r-1}}{n_r} + \dots + \frac{d_{-s}}{n_r\dots n_{s-1}} \leq \frac{n_r-1}{n_r} + \dots + \frac{n_{s-1}-1}{n_r\dots n_{s-1}} = 1 - \frac{1}{n_r\dots n_{s-1}} < 1$ for any $s \geq r$ we have $d_{-r} + x = d'_{-r} + y$ for some $0 \leq x, y < 1$ and thus it must hold that $d_{-r} = d'_{-r}$. Contradiction! ◀

▶ **Theorem 11.** *The class of torsion-free Abelian groups of rank 1 is regularly ω-automatic.*

More specifically, there is a parameterised ω-automatic presentation $\mathfrak{c}$, so that for all $(n_i)_{i\in\mathbb{N}}$ with $n_i \geq 2$ we have $\mathcal{S}(\mathfrak{c}[\operatorname{bin}(n_0)\#\operatorname{bin}(n_1)\#\dots]) \cong (\mathbb{Q}_c, +, <, \mathbb{Z})$ where $c = (c_p)_{p\in\mathbb{P}}$ with $\sum_{i=0}^{\infty} h_p(n_i) = c_p$ for all $p \in \mathbb{P}$.

Proof. It will be sufficient to construct a presentation $\mathfrak{c} = (\mathcal{A}, \mathcal{A}_+, \mathcal{A}_<, \mathcal{A}_{\mathbb{N}})$ for the subsemigroups $(\mathbb{Q}_c^+, +, <, \mathbb{N})$ of the semigroup $(\mathbb{Q}^+, +, <, \mathbb{N})$ of non-negative rational numbers,

because there is a first-order interpretation that interprets $(\mathbb{Q}_c, +, <, \mathbb{Z})$ in $(\mathbb{Q}_c^+, +, <, \mathbb{N})$ for all c.

For any $(n_i)_{i\in\mathbb{N}}$ that satisfies the conditions of the theorem every $r \in \mathbb{Q}_c^+$ has according to Lemma 10 a presentation of the form $\sum_{z=-l}^{k} d_z n_{\underline{z!}}$ for a unique coefficient sequence $(d_i)_{i=-l}^{k}$. Encode $(d_i)_{i=-l}^{k}$ as a word in the following way:

$$\begin{array}{llll} \mathrm{bin}(d_0) & \#\,\mathrm{bin}(d_1) & \#\ldots\#\,\mathrm{bin}(d_l) & \#\,\mathrm{bin}(d_{l+1}) \quad \ldots\#\,\mathrm{bin}(d_k)\#\square\ldots \\ \mathrm{bin}(d_{-1}) & \#\,\mathrm{bin}(d_{-2}) & \#\ldots\#\,\mathrm{bin}(d_{-l-1}) & \#\square\ldots \quad\ldots\square\ldots \\ \hline \mathrm{bin}(n_0) & \#\,\mathrm{bin}(n_1) & \#\ldots\#\,\mathrm{bin}(n_l) & \#\,\mathrm{bin}(n_{l+1}) \quad \ldots\#\,\mathrm{bin}(n_k)\#\ldots \end{array}$$

where the binary encoding of the numbers is padded by leading zeros so that the #'s of each row align with the #'s of the advice. With the advice string, the automaton $\mathcal{A}$ can verify that condition 1 of Lemma 10 holds for each triple $(\mathrm{bin}(d_i), \mathrm{bin}(d_{-i-1}), \mathrm{bin}(n_i))$ in a #-seperated segment and that $d_{-l} \neq 0 \vee (l = 1 \wedge d_{-l} = 0)$ and $d_k \neq 0 \vee k = 0 = d_k$ hold which ensures condition 2. The automaton $\mathcal{A}_{\mathbb{N}}$ merely has to check that the fractional part of r is zero, i.e. that the second row has the form $\mathrm{bin}(0)\#\square^\omega$. $\mathcal{A}_<$ recognises the length-lexicographical ordering on the digit sequences. To verify that addition is performed correctly the automaton has to check $\sum_{z=-l}^{k} d_z n_{\underline{z!}} + \sum_{z=-l'}^{k'} e_z n_{\underline{z!}} = \sum_{z=-l''}^{k''} s_z n_{\underline{z!}}$ given the encodings of the sequences $(d_z)_{-l\leq z\leq k}$, $(e_z)_{-l'\leq z\leq k'}$, and $(s_z)_{-l''\leq z\leq k''}$. This can be verified by computing the sum of every #-separated segment i modulo n_i while passing a carry bit from the lower significant segments to the higher significant segments. Note that $\frac{n_i+s_{-i-1}}{n_0 n_1 \cdots n_i} = \frac{1}{n_0 n_1 \cdot n_{i-1}} + \frac{s_{-i-1}}{n_0 n_1 \cdots n_i}$ and $(n_i + s_i) n_0 n_1 \ldots n_{i-1} = s_i n_0 n_1 \ldots n_{i-1} + n_0 \cdots n_i$. It is routine to construct an automaton that implements this idea on the described encoding. Note finally that the set of parameters $\{\mathrm{bin}(n_0)\#\mathrm{bin}(n_1)\#\ldots \mid n_i \geq 2\} = \{1\{0,1\}^+\#\}^\omega$ is clearly ω-regular. ◀

Next we show that every uniformly ω-automatic class $\mathcal{C}$ of Abelian groups has bounded rank. This also implies that every Abelian group that is ω-automatic with advice has finite rank. We need the following combinatorial fact about parameterised ω-automatic presentations.

▶ **Lemma 12.** *Let $\mathfrak{c}$ be a parameterised ω-automatic presentation. There is a constant $c \in \mathbb{N}$ such that whenever $\mathfrak{c}[\alpha]$ presents some countable structure $\mathfrak{A}$ for some advice α and f is a binary function of $\mathfrak{A}$ then for every substructure $\mathfrak{B} \subseteq \mathfrak{A}$ and for every finite subset C of $\mathfrak{B}$ there is a finite subset $D \supseteq C$ of $\mathfrak{B}$ with $|f(D,D)| \leq c \cdot |D|$.*

Proof. Due to Theorem 6 we can assume without loss of generality that $\mathfrak{c}$ is an injective presentation over finite words, so that C can be identified with a finite set over finite words, i.e. $C \subseteq \Sigma^*\{\square\}^\omega$ for a finite alphabet Σ. Let m be the maximal length of words in C. Then all words in C are $\sim_e^m$-equivalent. Let $C' \subseteq B$ be a $\sim_e^m$-class over B of maximal cardinality. Then there are ω-suffixes γ_0, γ_1 so that $C = C_0\gamma_0$ and $C' = C_1\gamma_1$ for some sets $C_0, C_1 \subseteq \Sigma_\square^m$. Let $D \coloneqq C \cup C'$. Then $f(D,D) = \bigcup_{i,j\in\{0,1\}} f(C_i\gamma_i, C_j\gamma_j)$. It suffices to show that $f(C_i\gamma_i, C_j\gamma_j)$ is contained in the union of no more than q $\sim_e^m$-equivalence classes over B. Due to the maximality of C' it then follows that $|f(C_i\gamma_i, C_j\gamma_j)| \leq q|C'| \leq q|D|$ for all $i,j \in \{0,1\}$ and thus $|f(D,D)| \leq 4q|D|$, so that $c \coloneqq 4q$ is the constant we are looking for. For this matter let $\mathcal{A}_f$ be the parameterised ω-automaton in the presentation that recognises the graph of f and let q be the number of states of $\mathcal{A}_f$. Towards a contradiction suppose there were $q+1$ words $\beta_1, \ldots, \beta_{q+1} \in f(C_i\gamma_i, C_j\gamma_j)$ that are pairwise not $\sim_e^m$-equivalent. By the pigeonhole principle there must then also be $i_0 \neq i_1$ with $\beta_{i_0} = f(c_{i_0}\gamma_i, c_{j_0}\gamma_j)$, $\beta_{i_1} = f(c_{i_1}\gamma_i, c_{j_1}\gamma_j)$ for some $c_{i_0}, c_{i_1} \in C_i$ and $c_{j_0}, c_{j_1} \in C_j$, so that $\mathcal{A}_f[\alpha]$ reaches the same state p after reading the m-prefix $\langle c_{i_0}, c_{j_0}, \beta_{i_0}\rangle[0,m)$ as after reading the m-prefix

$\langle c_{i_1}, c_{j_1}, \beta_{i_1}\rangle[0,m)$. Since $\mathcal{A}_f[\alpha]$ accepts $\langle c_{i_k}\gamma_i, c_{j_k}\gamma_j, \beta_k\rangle$ for $k \in \{0,1\}$ it accepts beginning in state p and from position m on the input tape both suffixes $\langle \gamma_i, \gamma_j, \beta_k\rangle[m,\infty)$ and thus both words $\langle c_{i_0}\gamma_i, c_{j_0}\gamma_j, \beta_0[0,m)\beta_k[m,\infty)\rangle$ for $k \in \{0,1\}$. Since f can map the pair $(c_i\gamma_i, c_j\gamma_j)$ only to one word, this implies $\beta_0[m,\infty) = \beta_1[m,\infty)$ in contradiction to $\beta_{i_0} \not\sim_e^m \beta_{i_1}$. ◀

Similar restrictions for classical automatic structures are well known and follow more or less directly from the pumping lemma for regular languages. However, in the presence of an advice string a pumping argument is not possible, because a manipulation of the elements of the structure via pumping would inevitably alter the advice. Therefore we need to employ a different combinatorial analysis, similar to techniques from [1].

We shall make use of Freiman's theorem, which has also been applied in the non-automaticity proof for $(\mathbb{Q},+)$ [23]. A *generalised arithmetic progression* P of rank $d \geq 1$ in a torsion-free Abelian group $(G,+)$ is a set of the form $P := \{a_0 + \sum_{i=1}^{d} z_i \cdot a_i : z_i \in \mathbb{Z}\}$ for $a_0, a_1, \ldots, a_d \in G$. A simplified version of Freiman's theorem reads as follows.

▶ **Theorem 13** (Freiman). *Let $(G,+)$ be a torsion-free Abelian group. There exists a function $r : \mathbb{Q}_c^+ \to \mathbb{N}$ such that for all finite subsets $A \subseteq G$ there is a generalised arithmetic progression P of rank $r\left(\frac{|A+A|}{|A|}\right)$ that contains A.*

▶ **Lemma 14.** *There exists a function $f : \mathbb{Q}^+ \to \mathbb{N}$ such that for every torsion-free Abelian group $(G,+)$ the following is true: For every finite set $X \subseteq G$ the rank of the subgroup generated by X is bounded by* $\operatorname{rank}(\langle X\rangle) \leq f\left(\frac{|X+X|}{|X|}\right)$.

Proof. Consider the torsion-free Abelian group $(\mathbb{Z}^\omega,+)$. By Freiman's Theorem there is a function $g : \mathbb{Q}^+ \to \mathbb{N}$ such that for every finite subset $X \subseteq \mathbb{Z}^\omega$ the following holds: Let $n := r\left(\frac{|X+X|}{|X|}\right)$, then $\langle X\rangle$ is contained in an n-dimensional generalised arithmetic progression $P = \{a_0 + k_1a_1 + \cdots + k_na_n \mid k_1, \ldots, k_n \in \mathbb{Z}\}$ for some $a_0, a_1, \ldots, a_n \in \mathbb{Z}^\omega$. Hence, $\langle X\rangle \subseteq \langle\{a_0, a_1, \ldots, a_n\}\rangle$ and therefore $\operatorname{rank}(\langle X\rangle) \leq \operatorname{rank}(\langle\{a_0, a_1, \ldots, a_n\}\rangle) \leq n+1$.

Let G be a torsion-free Abelian group and let X be finite subset of G. Then $\langle X\rangle_G$ is a finitely generated torsion-free Abelian group and therefore, by the classification of finitely generated Abelian groups, isomorphic to $(\mathbb{Z}^n,+)$ for some $n \in \mathbb{N}$. Consequently $\langle X\rangle$ is also isomorphic to a subgroup of $(\mathbb{Z}^\omega,+)$. Fix some embedding $\iota : \langle X\rangle \to (\mathbb{Z}^\omega,+)$. We can bound the rank of $\langle X\rangle_G$ by $\operatorname{rank}(\langle X\rangle) = \operatorname{rank}(\langle \iota(X)\rangle) \leq r\left(\frac{|X+X|}{|X|}\right) + 1 =: f\left(\frac{|X+X|}{|X|}\right)$. ◀

We are prepared to prove our claim. In fact, we show a slightly stronger result. A semigroup $\mathfrak{S}$ is cancellative if for all $x, y, z \in S$ it holds that $xy = xz$ implies $y = z$ and $yx = zx$ implies $y = z$. Every commutative cancellative semigroup $\mathfrak{S}$ can be embedded into an Abelian group in very much the same way as $(\mathbb{N},+)$ can be embedded into $(\mathbb{Z},+)$. More precisely there exists a unique Abelian group $G(\mathfrak{S})$ such that $\mathfrak{S}$ embeds into $G(\mathfrak{S})$ in the sense that whenever $\iota : \mathfrak{S} \to \mathfrak{G}$ is an embedding into some Abelian group $\mathfrak{G}$ then $\langle \iota(S)\rangle$ is isomorphic to $G(\mathfrak{S})$. For instance $G((\mathbb{N}^k,+)) \cong (\mathbb{Z}^k,+)$ for every $k \geq 1$ and $G(\mathfrak{G}) \cong \mathfrak{G}$ for all Abelian groups $\mathfrak{G}$. For more information we refer to [10].

▶ **Theorem 15.** *Let $\mathcal{C}$ be a uniformly ω-automatic class of countable commutative cancellative semigroups. Then the class $\mathcal{D} = \{G(\mathfrak{S}) \mid \mathfrak{S} \in \mathcal{C}\}$ has bounded rank.*

Proof. Let $\mathcal{C}$ be presented by a uniformly ω-automatic presentation $\mathfrak{c}$ over some parameter set P. Then let c be the constant from Lemma 12 with respect to $\mathfrak{c}$ and $+$, and let f be the function from Lemma 14. We claim that the rank of $\mathcal{D}$ is bounded by $f(c)$. Consider $\mathfrak{S} \in \mathcal{C}$ with $\mathfrak{G} := G(\mathfrak{S})$ and fix an embedding $\iota : \mathfrak{S} \to \mathfrak{G}$. Let $\mathfrak{H} \subseteq \mathfrak{G}$ be a free Abelian

subgroup of maximal rank. Then $\text{rank}(\mathfrak{G}) = \text{rank}(\mathfrak{H})$ and there is a subsemigroup $\mathfrak{T}$ of $\mathfrak{S}$ such that $\iota(T)$ generates $\mathfrak{H}$. In order to show that $\text{rank}(\mathfrak{H})$ is bounded by $f(c)$ it suffices to show that the rank of $\langle X \rangle$ is bounded by $f(c)$ for every finite subset X of $\mathfrak{H}$. So let X be a finite subset of $\mathfrak{H}$. Then there is a finite subset $U \subseteq T$ with $X \subseteq \langle \iota(U) \rangle$. By Lemma 12 there is a finite set V with $U \subseteq V \subseteq T$ such that $|V + V| \leq c|V|$. Hence $\text{rank}(\langle X \rangle) \leq \text{rank}(\langle \iota(V) \rangle) \leq f\left(\frac{|V+V|}{|V|}\right) \leq f(c)$. ◀

▶ **Corollary 16.** *The following classes are not uniformly ω-automatic:*

1. *The class $\{(\mathbb{Z}^n, +) \mid n \in N\}$ for any infinite set $N \subseteq \mathbb{N} \setminus \{0\}$. In particular the class of all free abelian groups is not uniformly ω-automatic.*
2. *The class $\{(\mathbb{N}^n, +) \mid n \in N\}$ for any infinite set $N \subseteq \mathbb{N} \setminus \{0\}$.*

4.3 Boolean Algebras

The previous results demonstrate that ω-automatic presentations with advice capture a much greater variety of structures than ordinary ω-automatic presentations. Moreover, uniformly ω-automatic classes can be surprisingly rich. On the other side we have also seen that (classes of) structures that are presentable in this way are still subject to certain restrictions. Therefore one might ask if there are also examples where we do not gain anything from the possibility to access an advice string or where the only uniformly ω-automatic classes are the trivial ones. In this section we will show that both is the case for the class of countable Boolean algebras. For the following we need the fact that every countable Boolean algebra is isomorphic to the interval algebra $\mathfrak{B}_{\mathfrak{L}}$ of a linear order $\mathfrak{L}$. The interval algebra of a linear order $\mathfrak{L}$ is the set algebra generated by the half-open intervals $\{[x, y) \mid x, y \in \mathfrak{L}, x < y\}$. A Boolean algebra $\mathfrak{B}$ is called *super-atomic* if for every element $b \in B \setminus \{\emptyset\}$ there is an atom $a \in B$ with $a \subseteq b$. Any super-atomic countable Boolean algebra is isomorphic to $\mathfrak{B}_\alpha$ for an ordinal α. For an introduction to the theory of Boolean algebras we refer to [9]. The automatic Boolean algebras have been fully classified in [15]: A countably infinite Boolean algebra is automatic if, and only if, it is isomorphic to $\mathfrak{B}_{\omega n}$ for some $n \geq 1$.

▶ **Theorem 17.** *A countably infinite Boolean algebra is ω-automatic with advice if, and only if, it is isomorphic to $\mathfrak{B}_{\omega n}$ for some natural number $n \geq 1$.*

Proof. Let $\mathfrak{B}$ be a countably infinite Boolean algebra that is not isomorphic to $\mathfrak{B}_{\omega n}$ for all $n \geq 1$. Suppose $\mathfrak{B}$ is ω-automatic with advice. Then $\mathfrak{B}$ has an injective ω-automatic presentation $\mathfrak{b}[\alpha]$ over finite words. We can add the length-lexicographical order to $\mathfrak{b}$ and obtain a presentation of $\mathfrak{B}^{\leq_{\text{llex}}} := (\mathfrak{B}, \leq_{\text{llex}})$ where $\leq_{\text{llex}}$ constitutes a linear order of order type ω on $\mathfrak{B}$. We show that one can construct from $\mathfrak{B}^{\leq_{\text{llex}}}$ an ω-automatic presentation with advice α of the ordinal ω^ω, which has already been shown not to be ω-automatic with advice in [13]. Note that ω^ω is isomorphic to the set $\{(n_0, \ldots, n_k) \in \mathbb{N}^* \mid n_k \neq 0\}$ with the length-lexicographical ordering on $\mathbb{N}^*$.

We construct an advice regular set P of pairwise disjoint elements of $\mathfrak{B}$ such that there are infinitely many $b \in \mathfrak{B}$ with $b \subseteq p$ for every $p \in P$. For the interval algebra of $(\mathbb{Q}, <)$, which is, up to isomorphism, the only countable atomless Boolean algebra, such a set P would be for example $P = \{[n, n+1) \mid n \in \mathbb{N}\}$. For $\mathfrak{B}_\alpha$ with $\alpha \geq \omega^2$ an example would be $P = \{[\omega n, \omega(n+1)) \mid n \in \mathbb{N}\}$. Therefore we can conclude that $\mathfrak{B}$ contains such a set because if $\mathfrak{B}$ contains an element $c \in B \setminus \{\emptyset\}$ such that there is no atom a with $a \subseteq c$, then the Boolean subalgebra $\{x \cap c \mid x \in B\}$ is an atomless countable Boolean algebra. As the example above shows, there is an infinite set P in $\mathfrak{B}$ with the described property. If $\mathfrak{B}$ contains no

such element, then $\mathfrak{B}$ is super-atomic and therefore isomorphic to $\mathfrak{B}_\alpha$ for an ordinal $\alpha \geq \omega^2$, in which case there is also such a P.

It remains to show how to obtain such a set P that is ω-regular with advice α. Consider the formula $\psi_P(x,y) := ((x = y \vee x \cap y = \emptyset) \wedge \exists^\infty z(z \subseteq x) \wedge \exists^\infty z(z \subseteq y))$. Then $\mathfrak{B} \models \exists^{2\text{-ram}} xy \psi_P(x,y)$ and an application of Lemma 7 to the relation defined by $\psi_P(x,y)$ yields an infinite set P with $\mathfrak{B} \models \psi_P(a,b)$ for all $a, b \in P$ that is ω-regular with advice α and has the required property. Let $p(i)$ denote the i-th element of P in the well-order $\leq_{\text{llex}}$ and for every $p \in P$ let $a_p(i)$ be the i-th element below p with respect to $\leq_{\text{llex}}$ (here we have to omit the empty set in the enumeration of the elements below p). We encode a sequence $(n_0, n_1, \ldots, n_k)$ as the element $\bigcup_{0 \leq i \leq k} a_{p(i)}(n_i)$. Thus for example the sequence $(2,0,1)$ is encoded by the element $a_{p(0)}(2) \cup a_{p(1)}(0) \cup a_{p(2)}(1)$. Note that the i-th component of the sequence encoded by an element m can be retrieved from m as $p(i) \cap m$ and the length of a sequence is determined by the greatest i such that $p(i) \cap m \neq \emptyset$. It is now a simple exercise to construct a first-order interpretation of ω^ω. ◀

The next step is to ask which classes of countable Boolean algebras are uniformly ω-automatic. Trivially this is the case if the class is a finite collection of Boolean algebras of the form $\mathfrak{B}_{\omega n}$. We show that these are indeed the only examples. In the following we find it easier to work with a slightly different view on the structure of a Boolean algebra $\mathfrak{B}_{\omega n}$. Note that $\mathfrak{B}_\omega$ is isomorphic to the set algebra of all finite and co-finite subsets of ω, that is on $\mathcal{P}_{\text{fc}}(\omega) = \{X \subseteq \omega \mid X \text{ or } \omega \setminus X \text{ is finite}\}$, and that $\mathfrak{B}_{\omega n} \cong \mathfrak{B}_\omega^n$ for all $n \geq 1$.

▶ **Lemma 18.** *There exists an* FOC*-interpretation $\mathcal{I}$ such that the following holds: If $\mathfrak{A} \cong (\mathfrak{B}_\omega^n, P, \preceq)$, i.e. $\mathfrak{A}$ is isomorphic to the n-fold product of the Boolean algebra $\mathfrak{B}_\omega$, expanded by the unary relation $P = \{(\omega, \emptyset, \ldots, \emptyset), (\emptyset, \omega, \emptyset, \ldots, \emptyset), \ldots, (\emptyset, \ldots, \emptyset, \omega)\}$ and a linear order $\preceq$ of order-type ω, then $\mathcal{I}(\mathfrak{A}) \cong (\mathbb{N}^n, +)$.*

Proof. It is not hard to see that $(\mathbb{N}, +)$ is FOC-interpretable in $(\mathfrak{B}_\omega, P, \preceq)$. Without loss of generality we can assume that $\preceq$ behaves on the atoms like the natural linear order, that is $\{0\} \prec \{1\} \prec \cdots$. The idea is to identify the finite sets in $\mathfrak{B}_\omega$ with the binary expansions of natural numbers. Accordingly, a finite set X presents the number $n(X) = \sum_{i \in X} 2^i$. The domain formula has to express that x is a finite set. This can be done by the formula $\delta'(x) := \neg\exists^\infty y(y \subseteq x)$. It is routine to construct a formula $\varphi'_+(x,y,z)$ such that for all finite sets $X, Y, Z \subseteq \omega$ it holds that $(\mathfrak{B}_\omega, P, \preceq) \models \varphi_+(X,Y,Z)$ if, and only if, $n(X) + n(Y) = n(Z)$. We describe how to transform this interpretation into an interpretation $\mathcal{I}$ that interprets $(\mathbb{N}^n, +)$ in $(\mathfrak{B}_\omega^n, P, \preceq)$ for all $n \geq 1$. The idea for the general case is to use the predicate P to perform addition in every component separately. An element of $(m_1, \ldots, m_n)$ is now encoded by the tuple $(X_1, \ldots, X_n)$ of finite sets with $n(X_i) = m_i$ for all $1 \leq i \leq n$. The correctness of the addition now has to be checked for every component separately. Accordingly we define $\delta(x) \equiv \forall p \in P : \delta''(x \cap p, p)$ and $\varphi_+(x,y,z) \equiv \forall p \in P : \varphi''_+(x \cap p, y \cap p, z \cap p, p)$ where $\varphi''_{\mathbb{N}}$ and δ'' are obtained from δ' and φ'_+ by restricting all quantifications to elements below p. ◀

Next we show that the predicate P is definable in $(\mathfrak{B}_\omega^n, \preceq)$.

▶ **Lemma 19.** *Let $(X_{ij})_{1 \leq i,j \leq n}$ be a collection of finite subsets of ω such that for all $i \in \{1, \ldots, n\}$:*

- *$X_{ki} \cap X_{\ell i} = \emptyset$ for all $1 \leq k < \ell \leq n$ with $k, \ell \neq i$ and*
- *$X_{ii} = \bigcup_{j \neq i} X_{ji}$.*

Then there is an automorphism of $\mathfrak{B}_\omega^n$ which maps the n-tuple

$$((X_{i1}, \ldots, X_{i(i-1)}, \omega \setminus X_{ii}, X_{i(i+1)}, \ldots, X_{in}))_{1 \leq i \leq n}$$

of elements of $\mathcal{P}_{fc}(\omega)^n$ to the tuple $((\omega, \emptyset, \ldots, \emptyset), (\emptyset, \omega, \emptyset, \ldots, \emptyset), \ldots, (\emptyset, \ldots, \emptyset, \omega))$.

▶ **Lemma 20.** *There is an* FOC*-interpretation $\mathcal{J}$ such that the following is true: If a structure $\mathfrak{A}$ is isomorphic to $(\mathfrak{B}_\omega^n, \leq)$, where $\leq$ is a linear order on $\mathcal{P}_{fc}(\omega)^n$ of order type ω then the structure $\mathcal{J}(\mathfrak{A})$ is isomorphic to a structure $(\mathfrak{B}_\omega^n, P, \preceq)$, where P and $\preceq$ are as in Lemma 18.*

Proof. It suffices to show that a set of the form $\{(X_{i1}, \ldots, X_{i(i-1)}, \omega \setminus X_{ii}, X_{i(i+1)}, \ldots, X_{in}) \mid 1 \leq i \leq n\}$, where $(X_{ij})_{1 \leq i,j \leq n}$ are as described in Lemma 19, is definable by an FOC-formula.

First, we define the elements of $\mathfrak{B}_\omega^n$ that are finite in all but exactly one component. This is done by the following formula $\mathrm{Comp}(x) := \exists^\infty z(z \subseteq x) \wedge \neg\exists y(y \subseteq x \wedge \exists^\infty z(z \subseteq y) \wedge \exists^\infty z(z \subseteq x \setminus y))$, which states that x is infinite and there is no infinite subset y of x such that $x \setminus y$ is also infinite. This ensures that $x = (X_1, \ldots, X_{i-1}, \omega \setminus X_i, X_{i+1}, \ldots, X_n)$ for some finite sets $X_1, \ldots, X_n \subseteq \omega$ and some $i \leq n$.

Next we employ the linear order to preselect n such elements which are infinite in pairwise different components. $\mathrm{Sel}(x) := \mathrm{Comp}(x) \wedge \forall y((\mathrm{Comp}(y) \wedge y < x) \to \neg\exists^\infty z(z \subseteq x \cap y))$. The elements of Sel are not yet of the type that we need. First, the elements of Sel might have finite intersections and second there might be finitely many atoms that are not below any element of Sel. Therefore we need to modify the elements of Sel so that they are disjoint and every atom is below one of these elements. This, however, can easily be achieved by a first-order formula. ◀

▶ **Corollary 21.** *Let $\mathcal{C}$ be a class of countably infinite Boolean algebras. Then the following three conditions are equivalent:*

1. *$\mathcal{C}$ is uniformly automatic.*
2. *$\mathcal{C}$ is uniformly ω-automatic.*
3. *$\mathcal{C} \cong \{\mathfrak{B}_{\omega n} \mid n \in N\}$ for some finite set $N \subseteq \mathbb{N} \setminus \{0\}$.*

Proof. Clearly (1) implies (2) and (3) implies (1) because every Boolean algebra of the form $\mathfrak{B}_{\omega n}$ is automatic and every finite class of automatic structures is uniformly automatic.

It remains to prove that (2) implies (3). Let $\mathcal{C}$ be a uniformly ω-automatic class of countably infinite Boolean algebras. Then every $\mathfrak{B} \in \mathcal{C}$ is ω-automatic with advice and hence isomorphic to some $\mathfrak{B}_{\omega n}$. Consequently $\mathcal{C} \cong \{\mathfrak{B}_{\omega n} \mid n \in N\}$ for some set $N \subseteq \mathbb{N} \setminus \{0\}$. Suppose that N is infinite. By Theorem 6 there is an injective uniformly ω-automatic presentation $\mathfrak{c}$ of $\mathcal{C}$ over finite words. We can now expand $\mathfrak{c}$ by the length lexicographical order and use the interpretations $\mathcal{I}$ and $\mathcal{J}$ from Lemma 18 and Lemma 20 to obtain a uniformly ω-automatic presentation of the class $\{(\mathbb{N}^n, +) \mid n \in N\}$, contradicting Corollary 16. ◀

4.4 Monoids, Groups, and Integral Domains

The ideas and methods presented in the previous sections are powerful enough to provide several other non-automaticity results. In particular it can be shown that:

- The free semigroup with two generators is not a substructure of any countable structure that is ω-automatic with advice.
- $(\mathbb{N}, \cdot)$ is not a substructure of any countable structure that is ω-automatic with advice.
- No infinite integral domain has an injective ω-automatic presentation with advice.
- The field of reals is not ω-automatic with advice.

Due to space constraints we have to abstain from presenting the proofs here. Instead we refer the interested reader to the upcoming full version of this paper.

5 Closure Properties and Counting Problems

It is easy to see that the class of automatic structures is closed under the standard composition operators such as disjoint union and direct product and that the same constructions still work in the presence of an advice. However, when we consider uniformly automatic classes then the closure under such compositions is a much stronger notion because compositions of arbitrary width must be presented uniformly. It is therefore not surprising that the situation becomes more diverse in this setting.

▶ **Definition 22.** Let $\mathcal{C}$ be a class of τ-structures. Then $\mathcal{C}^\times$ denotes the closure of $\mathcal{C}$ under direct products and, in case that τ is relational, $\mathcal{C}^\uplus$ denotes the closure of $\mathcal{C}$ under disjoint unions. That is $\mathcal{C}^\times = \{\mathfrak{A}_1 \times \cdots \times \mathfrak{A}_n \mid n \geq 1, \mathfrak{A}_1, \ldots, \mathfrak{A}_n \in \mathcal{C}\}$ and $\mathcal{C}^\uplus = \{\mathfrak{A}_1 \uplus \cdots \uplus \mathfrak{A}_n \mid n \geq 1, \mathfrak{A}_1, \ldots, \mathfrak{A}_n \in \mathcal{C}\}$.

It is not hard to see that uniformly (ω-)tree-automatic classes behave very well under the two closure operators that we defined above.

▶ **Lemma 23.** *Let $\mathcal{C}$ be a uniformly (ω-)tree-automatic class of structures. From a given (ω-)tree-automatic presentation $(P, \mathfrak{c})$ of $\mathcal{C}$ one can effectively construct (ω-)tree-automatic presentations $(P^\times, \mathfrak{c}^\times)$ of $\mathcal{C}^\times$, and $(P^\uplus, \mathfrak{c}^\uplus)$ of $\mathcal{C}^\uplus$. Moreover, the regularity of the advice set is preserved.*

For uniformly (ω-)automatic classes we do not enjoy the same closure properties as in the tree case.

▶ **Corollary 24.** *There is a regularly automatic class $\mathcal{C}$ such that the closure under direct products $\mathcal{C}^\times$ is not uniformly ω-automatic.*

Proof. The free Abelian groups of finite rank are up to isomorphism the finite direct products of the automatic structure $(\mathbb{Z}, +)$. Hence by Corollary 16, $\{(\mathbb{Z}, +)\}^\times$ is not uniformly ω-automatic. ◀

However, if we restrict ourselves to classes of finite structures then uniformly automatic classes are equally well behaved.

▶ **Lemma 25.** *Let $\mathcal{C}$ be a uniformly automatic class of* finite *structures. From a given automatic presentation $(P, \mathfrak{c})$ of $\mathcal{C}$ one can effectively construct a uniformly automatic presentation $(P^\times, \mathfrak{c}^\times)$ of $\mathcal{C}^\times$ and $(P^\uplus, \mathfrak{c}^\uplus)$ of $\mathcal{C}^\uplus$. Moreover, regularity of the advice set is preserved.*

Product closures play an important role in the structure theory of Abelian groups, since classification theorems in that domain often take the form: $\mathfrak{G}$ is a group with property Prop, if, and only if, $\mathfrak{G}$ is isomorphic to a direct sum of groups from a class $\mathcal{C}_{\text{Prop}}$; where $\mathcal{C}_{\text{Prop}}$ is a class of prime groups with property Prop, in the sense that they have no non-trivial decomposition into groups with property Prop and that $\mathcal{C}_{\text{Prop}}^\times = \mathcal{C}$.

The *divisible Abelian groups* are an example for a class that has a classification theorem of this form. An Abelian group $\mathfrak{G}$ is called *divisible*, if for every $g \in G$ and $n \in \mathbb{N}$ there is a $h \in G$ with $n \cdot h = g$. The prime groups in the classification theorem for divisible Abelian groups are $(\mathbb{Q}, +)$ and the Prüfer p-groups $\mathbb{Z}(p^\infty)$, which are isomorphic to $\mathbb{Q}_{\chi_p}/\ \mathbb{Z}$ where $\chi_p = (\chi_p)_{q \in \mathbb{P}}$ is the sequence with $(\chi_p)_p = \infty$ and $(\chi_p)_q = 0$ for $p \neq q$. Let us use in general the notation $\mathbb{Z}(n^\infty)$ for the subgroup of $\mathbb{Q}/\ \mathbb{Z}$ that is generated by $\{\frac{1}{n^k} \mid k \in \mathbb{N}\}$. Note that $\{\mathbb{Z}(p^\infty) \mid p \in \mathbb{P}\}^\times = \{\mathbb{Z}(n^\infty) \mid n \geq 2\}$ thus the countable divisible Abelian groups are the product closure of the class $\mathcal{C}_{\text{Div}} = \{(\mathbb{Q}, +)\} \cup \{\mathbb{Z}(n^\infty) \mid n \geq 2\}$.

Note that from the regularly ω-automatic presentation of $(\mathbb{Q}_c, +, <, \mathbb{Z})$ in Theorem 11 we can obtain a regularly automatic presentation of the subgroups of $\mathbb{Q}/\mathbb{Z}$ via a first-order interpretation. This is a parameterised automatic presentation of $\mathcal{C}_{\mathrm{Div}}$ with parameter set $P = \{\mathrm{bin}(2)\#\,\mathrm{bin}(3)\ldots\} \cup \{(\mathrm{bin}(n)\#)^\omega : n \geq 2\}$. Lemma 23 yields an ω-tree automatic presentation of $\mathcal{C}^\times_{\mathrm{Div}}$ with parameter set $P^\times$, which has a decidable MSO-theory.

▶ **Corollary 26.** *The class of countable divisible Abelian groups is strongly ω-tree-automatic.*

Even in cases when a class of structures does not admit a regular presentation up to isomorphism (possibly already due to cardinality reasons), a regular presentation up to elementary equivalence may still be possible, and we would thereby still get a decision procedure for the first-order theory of the class. This is in fact also possible for the class of all Abelian groups itself, whereby we can reprove Szmielew's decidability result for the first-order theory of Abelian groups [21]. We rely here on the fact that every Abelian group is elementary equivalent to a Szmielew group and every Szmielew group is isomorphic to a countably infinite direct sum of subgroups of $\mathbb{Q}$ and $\mathbb{Q}/\mathbb{Z}$ [11]. From this fact and Lemma 23 we obtain the following corollary.

▶ **Corollary 27.** *There is a regularly ω-tree-automatic presentation of the class of all Abelian groups up to elementary equivalence.*

At last, let us mention an application that arises from the semantic shift in the transition between automatic presentations with advice and uniformly automatic presentations. In model theory one is often interested in the number of isomorphism types of structures from a given class that satsify a given formula. It turns out that counting problems of this kind can also be handled in our framework if the presentation of the class fulfills the following uniqueness condition: An [ω-](tree-)automatic presentation $\mathfrak{c}$ of a class $\mathcal{C}$ has the **unique representation property** if for any $\mathfrak{A} \in \mathcal{C}$ there is exactly one parameter α such that $\mathcal{S}(\mathfrak{c}[\alpha]) \cong \mathfrak{A}$.

▶ **Lemma 28.** *If a class $\mathcal{C}$ has a regularly (ω-)automatic presentation $\mathfrak{c}$ with the unique representation property then there is a uniform decision procedure to determine, for given $\varphi \in \mathrm{FO}$ and $k, m \in \mathbb{N}$, whether the number of pairwise non-isomorphic models of φ in $\mathcal{C}$ is a finite number n with $n \equiv k \mod m$, whether it is at most countably infinite, or whether it is uncountable.*

Proof. We can construct from $\mathfrak{c}$ an automatic presentation of $\mathfrak{A}_\mathcal{C} = ((\uplus_{\alpha\in P}\mathcal{S}(\mathfrak{c}[\alpha]))\cup\{\mathcal{S}(\mathfrak{c}[\alpha]) \mid \alpha \in P\}, \sim)$, where $\mathcal{S}(\mathfrak{c}[\alpha]) \sim a$ holds if $a \in \mathcal{S}(\mathfrak{c}[\alpha])$. Let ψ be the formula obtained from φ by relativising all quantifiers Qx by $a \sim x$. Because of the unique representation property the decision problems reduce to checking whether $\mathfrak{A}_\mathcal{C}$ satisfies $\exists^{(k,m)}a\psi$, $\exists^{\leq\aleph_0}a\psi$, or $\exists^{>\aleph_0}a\psi$, respectively. ◀

Unfortunately even in the simplest conceivable case, that is regularly automatic classes of finite sets, the unique representation property is undecidable.

▶ **Theorem 29.** *The problem to decide whether a regularly automatic presentation has the unique representation property is Π^0_1-complete for the empty signature and for signatures with only monadic predicates (even for classes of finite structures), and Π^1_1-hard for any signature with at least one predicate of arity at least two.*

References

1. Faried Abu Zaid, Erich Grädel, and Łukasz Kaiser. The Field of Reals is not omega-Automatic. In Christoph Dürr and Thomas Wilke, editors, *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012*, 2012. doi:10.4230/LIPIcs.STACS.2012.577.
2. Reinhold Baer. Abelian groups without elements of finite order. *Duke Mathematical Journal*, 3:68–122, 1937. doi:10.1215/S0012-7094-37-00308-9.
3. V. Bárány, E. Grädel, and S. Rubin. Automata-based presentations of infinite structures. In *Finite and Algorithmic Model Theory*, London Mathematical Society Lecture Note Series (No. 379). Cambridge University Press, 2011. doi:10.1017/CBO9780511974960.002.
4. Achim Blumensath. Automatic Structures. Diploma thesis, RWTH-Aachen, 1999. URL: http://www.logic.rwth-aachen.de/pub/blume/AutStr.ps.gz.
5. Achim Blumensath and Erich Grädel. Finite Presentations of Infinite Structures: Automata and Interpretations. *Theory of Computing Systems*, 37:641–674, 2004. doi:10.1007/s00224-004-1133-y.
6. Christian Choffrut and Serge Grigorieff. Uniformization of rational relations. In *Jewels Are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 59–71. Springer-Verlag, 1999. doi:10.1007/978-3-642-60207-8_6.
7. Thomas Colcombet and Christof Löding. Transforming structures by set interpretations. *Logical Methods in Computer Science*, 3(2), 2007. doi:10.2168/LMCS-3(2:4)2007.
8. László Fuchs. *Infinite Abelian Groups*, volume Bd. 1. Academic Press, 1970. doi:10.1007/978-3-319-19422-6.
9. Sergey Goncharov. *Countable Boolean Algebras and Decidability*. Siberian School of Algebra and Logic. Springer, 1997.
10. P. A. Grillet. *Commutative Semigroups*. Advances in Mathematics. Springer US, 2004. doi:10.1007/978-1-4757-3389-1.
11. Wilfrid Hodges. *Model Theory*. Number 42 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1993. doi:10.1017/CBO9780511551574.
12. Łukasz Kaiser, Sasha Rubin, and Vince Bárány. Cardinality and counting quantifiers on omega-automatic structures. In Susanne Albers and Pascal Weil, editors, *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science, STACS 2008*, pages 385–396, 2008. doi:10.4230/LIPIcs.STACS.2008.1360.
13. Alexander Kartzow and Philipp Schlicht. Structures without scattered-automatic presentation. In Paola Bonizzoni, Vasco Brattka, and Benedikt Löwe, editors, *The Nature of Computation. Logic, Algorithms, Applications*, volume 7921 of *Lecture Notes in Computer Science*, pages 273–283. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-39053-1_32.
14. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Selected Papers from the International Workshop on Logical and Computational Complexity*, LCC'94, pages 367–392. Springer-Verlag, 1995.
15. Bakhadyr Khoussainov, Sasha Rubin, A. Nies, and F. Stephan. Automatic structures: Richness and limitations. In *LICS 2004*, pages 44–53, 2004. doi:10.1109/LICS.2004.1319599.
16. Alex Kruckman, Sasha Rubin, John Sheridan, and Ben Zax. A Myhill-Nerode theorem for automata with advice. *Electronic Proceedings in Theoretical Computer Science*, 96:238–246, October 2012. doi:10.4204/EPTCS.96.18.
17. Dietrich Kuske. Is Ramsey's theorem omega-automatic? In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science*, volume 5 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 537–548. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010. doi:10.4230/LIPIcs.STACS.2010.2483.

18 Dietrich Kuske, Jiamou Liu, and Markus Lohrey. The isomorphism problem on classes of automatic structures. In *LICS 2010*, pages 160–169, 2010. doi:10.1090/S0002-9947-2013-05766-2.

19 André Nies. Describing groups. *Bulletin of Symbolic Logic*, 13(3):305–339, September 2007. doi:10.2178/bsl/1186666149.

20 Sasha Rubin. Automata presenting structures: A survey of the finite string case. *Bulletin of Symbolic Logic*, 14(2):169–209, 2008. doi:10.2178/bsl/1208442827.

21 Wanda Szmielew. Elementary properties of abelian groups. *Fundamenta Mathematicae*, 41(2):203–271, 1955.

22 Simon Thomas. The classification problem for torsion-free abelian groups of finite rank. *J. Amer. Math. Soc*, 16:233–258, 2001.

23 Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *J. Symb. Log.*, 76(4):1341–1351, 2011. doi:10.2178/jsl/1318338853.

A Proofs Omitted from Section 3

▶ **Theorem 6.** *A class $\mathcal{C}$ of countable structures has a parameterised ω-automatic presentation $\mathfrak{c}$ with parameter set P, if, and only if, it has an injective parameterised ω-automatic presentation over finite words $\mathfrak{c}'$ with the same parameter set P. Moreover $\mathfrak{c}'$ can be effectively constructed from $\mathfrak{c}$.*

Proof. Let $\mathfrak{c} = (\mathcal{A}, \mathcal{A}_\approx, (\mathcal{A}_R)_{R\in\tau})$ be the given presentation with advice alphabet Γ. We slightly modify the presentation in such a way that $\mathfrak{d}[\alpha]$ is a well-defined structure for every $\alpha \in \Gamma^\omega$. For this matter replace $\mathcal{A}_R$ by an automaton $\mathcal{A}'_R$ with $L(\mathcal{A}'_R) = \{\overline{a} \otimes \alpha \in L(\mathcal{A}_R) | \overline{a} \in L(\mathcal{A}[\alpha])^r\}$ which ensures that $\mathcal{A}'_R[\alpha]$ recognises a relation on the domain $L(\mathcal{A}[\alpha])$ for every $\alpha \in \Gamma^\omega$. To ensure that $\approx_\alpha$ is always a congruence relation, we replace $\mathcal{A}_\approx$ by an automaton $\mathcal{A}'_\approx$ so that $\mathcal{A}'_\approx[\alpha]$ recognises $L(\mathcal{A}_\approx[\alpha])$ if this is a congruence relation, and the equality relation otherwise. Since the property that $\approx_\alpha$ is a congruence relation is first-order definable, it is a property that can be checked by an automaton, i.e. such an $\mathcal{A}'_\approx$ can indeed be constructed. Define now the structure $\mathcal{S}_\approx(\mathfrak{c})$ as the disjoint union of all $\mathcal{S}_\approx(\mathfrak{c}[\alpha])$ for $\alpha \in \Gamma^\omega$. Then $(\mathcal{A}, \mathcal{A}'_\approx, (\mathcal{A}'_R)_{R\in\tau})$ is an ω-automatic presentation of $\mathcal{S}_\approx(\mathfrak{c})$. In the following we makes use of [12, Proposition 3.1] which gives a criterion for countability of sets that are defined by a formula $\varphi(x, \overline{z})$ with parameters $\overline{z}$ in an ω-automatic structure. We apply the criterion to the formula $\varphi(x, z) \coloneqq x \sim z$ and the $\tau \uplus \{\sim\}$-structure that we get from $\mathcal{S}_\approx(\mathfrak{c})$ by enlarging the universe with the elements α for every $\alpha \in \Gamma^\omega$ and defining the relation $\sim$ on the enlarged universe via $w \sim \alpha :\Leftrightarrow \alpha \in \Gamma^\omega$ and $w \in \mathcal{S}_\approx(\mathfrak{c}[\alpha])$. It is easy to see that $\mathfrak{A}_\mathfrak{c}$ is still an ω-automatic structure. Since $\varphi(-, \alpha)$ defines a countable set in $\mathfrak{c}$ for every $\alpha \in P$ the proposition thus implies that that there is a constant c computable from $\mathfrak{c}$ such that the formula

$$\psi(z, x_1, \dots, x_c) \coloneqq \bigwedge_{1\le i\le c} x_i \sim z \wedge \forall x(x \sim z \to \exists y(\bigvee_{1\le i\le c} y \sim_e x_i \wedge y \approx x))$$

defines an ω-automatic relation $R_\psi \subseteq \Gamma^\omega \times (\Sigma^\omega)^c$ in $\mathfrak{A}_\mathfrak{c}$ with $\alpha R_\psi \neq \emptyset$ for every $\alpha \in P$. By the uniformisation theorem for ω-automatic relations [6], there is an ω-automatic function $f_{R_\psi} : R_\psi(\Sigma^\omega)^c \to (\Sigma^\omega)^c$ with $f_{R_\psi}(\alpha) \in \alpha R_\psi$ for all $\alpha \in R_\psi(\Sigma^\omega)^c$.

We are now prepared to construct $\mathfrak{d}'$. Intuitively we are going to use f_{R_ψ} to pick $x_1, \dots, x_c$ from the original presentation such that every element has a $\approx$-representative in $L \coloneqq L(\mathcal{A}[\alpha]) \cap \bigcup_{1\le i\le c}[x_i]_{\sim_e}$. Then for every $y \in L$ we just cut y from the point where it coincides with some x_i and annotate the resulting string with the respective end-class. More

formally we first expand the alphabet Σ by new symbols $\{1, \dots, c\}$. The domain automaton is constructed from the formula:

$$\varphi_A(\alpha, x) := \exists y \in \Sigma^*, i \in \{1, \dots, c\}(x = yi\square^\omega \wedge (y(f_{R_\psi}(\alpha)_i[|y|, \infty)) \in L(\mathcal{A}[\alpha])).$$

Similarly for $S \in \{\approx\} \cup \tau$ we construct an automaton by the formula

$$\varphi_S(\alpha, y_1 i_1 \square^\omega, \dots, y_k i_k \square^\omega) := S(y_1(f_{R_\psi}(\alpha)_{i_1}[|y_1|, \infty)), \dots, y_k(f_{R_\psi}(\alpha)_{i_k}[|y_k|, \infty))).$$

The corresponding relations can easily be recognised by Muller automata. The presentation can be made injective by taking the length-lexicographic smallest representative of any $\approx$-class. ◀

B Proofs Omitted from Section 4

▶ **Lemma 19.** *Let $(X_{ij})_{1 \le i,j \le n}$ be a collection of finite subsets of ω such that for all $i \in \{1, \dots, n\}$:*

- *$X_{ki} \cap X_{\ell i} = \emptyset$ for all $1 \le k < \ell \le n$ with $k, \ell \neq i$ and*
- *$X_{ii} = \bigcup_{j \neq i} X_{ji}$.*

Then there is an automorphism of $\mathfrak{B}_\omega^n$ which maps the n-tuple

$$((X_{i1}, \dots, X_{i(i-1)}, \omega \setminus X_{ii}, X_{i(i+1)}, \dots, X_{in}))_{1 \le i \le n}$$

of elements of $\mathcal{P}_{fc}(\omega)^n$ to the tuple $((\omega, \emptyset, \dots, \emptyset), (\emptyset, \omega, \emptyset, \dots, \emptyset), \dots, (\emptyset, \dots, \emptyset, \omega))$.

Proof. The automorphism is constructed as follows: For all $1 \le i \le n$ fix a bijection π_i between the atoms below $(X_{i1}, \dots, X_{i(i-1)}, \omega \setminus X_{ii}, X_{i(i+1)}, \dots, X_{in})$ and the atoms below $(\emptyset, \dots, \emptyset, \underbrace{\omega}_{\text{position } i}, \emptyset, \dots, \emptyset)$. Because of the properties of $(X_{ij})_{1 \le i,j \le n}$, every atom appears in the domain and the range of exactly one π_i. Thus, we can combine $\pi_1, \dots, \pi_n$ to a permutation $\pi = \bigcup_{1 \le i \le n} \pi_i$ on the atoms of $\mathfrak{B}_\omega^n$. We lift π to a permutation ρ on $\mathcal{P}(\omega)^n$ by

$$\rho((X_1, \dots, X_n)) = \bigcup_{a \text{ atom below } (X_1, \dots, X_n)} \pi(a).$$

Then ρ is an automorphism on $(\mathcal{P}(\omega), \cup, \cap, \omega, \emptyset)^n$ because ρ is derived from a permutation of the atoms. Every such permutation is an automorphism because $(\mathcal{P}(\omega), \cup, \cap, \omega, \emptyset)^n$ is isomorphic to $(\mathcal{P}(\omega), \cup, \cap, \omega, \emptyset)$ via the automorphism

$$(X_1, \dots, X_n) \mapsto \bigcup_{1 \le i \le n} \{na + (i-1) \mid a \in X_i\}.$$

Further, the restriction of ρ to $\mathcal{P}_{\text{fc}}(\omega)^n$ is a permutation on $\mathcal{P}_{\text{fc}}(\omega)^n$, hence ρ is an automorphism on $\mathfrak{B}_\omega^n \subseteq (\mathcal{P}(\omega), \cup, \cap, \omega, \emptyset)^n$. ◀

C Proofs Omitted from Section 5

▶ **Lemma 23.** *Let $\mathcal{C}$ be a uniformly (ω-)tree-automatic class of structures. From a given (ω-)tree-automatic presentation $(P, \mathfrak{c})$ of $\mathcal{C}$ one can effectively construct (ω-)tree-automatic presentations $(P^\times, \mathfrak{c}^\times)$ of $\mathcal{C}^\times$, and $(P^\uplus, \mathfrak{c}^\uplus)$ of $\mathcal{C}^\uplus$. Moreover, the regularity of the advice set is preserved.*

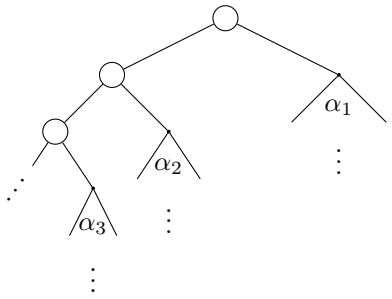


Figure 1 The Parameters for the class $\mathcal{C}^\times$.

Proof. Construction of $(P^\times, \mathfrak{c}^\times)$: Suppose $\mathcal{C}$ is presented by the uniform (ω-)tree-automatic presentation $\mathfrak{c}$ over the advice set P. As the construction is rather straightforward we only give the parameter set for the presentation and the idea for the encoding. The parameter set consists of all trees where the right child of every node in the left-most branch induces a subtree which is in P.

This is depicted in Figure 1 . Such an advice presents the structure $\mathcal{S}(\mathfrak{c}[\alpha_1]) \times \mathcal{S}(\mathfrak{c}[\alpha_2]) \times \cdots \times \mathcal{S}(\mathfrak{c}[\alpha_n])$. Let $t_1, \ldots, t_n$ be elements of $\mathcal{S}(\mathfrak{c}[\alpha_1]), \ldots, \mathcal{S}(\mathfrak{c}[\alpha_n])$, respectively. Then the element $(t_1, \ldots, t_n)$ is put together in the same way as the advices.

Construction of $(P^\uplus, \mathfrak{c}^\uplus)$: We use the same advice set as in (1). The elements are encoded by trees where all except one node 0^n1 of the form $\{0\}^*1$ are leafs labelled with a new dummy symbol and 0^n1 induces a subtree t such that $t \in L(\mathcal{A}[\alpha_n])$ ◀

▶ **Lemma 30.** *Let $\mathcal{C}$ be a regularly automatic class of finite structures. Then there is a presentation $\mathfrak{c} = (\mathcal{A}, (\mathcal{A}_R)_{R\in\tau})$ with regular parameter set P such that $|w| = |\alpha|$ for all $w \in L(\mathcal{A}[\alpha])$.*

Proof. Let $\mathfrak{c}' = (\mathcal{A}', (\mathcal{A}'_R)_{R\in\tau})$ be an injective uniform automatic presentation of $\mathcal{C}$ where the regular parameter set P' is recognised by $\mathcal{A}'_p$. Since $\mathfrak{c}$ is injective and $\mathcal{C}$ is a class of finite structures, $L(\mathcal{A}'[\alpha])$ is finite for all $\alpha \in L(\mathcal{A}'_p)$. Extend the alphabet of the presentation by a new padding symbol $\#$ and consider the injective mapping $\pi : L(\mathcal{A}'_p) \to L(\mathcal{A}'_p)\#^*$ given by $\alpha \mapsto \alpha\#^k$ with $k = \max\{0, \max\{|w| - |\alpha| \mid \alpha \otimes w \in L(\mathcal{A}'_p)\}\}$. The language $P := \pi(L(\mathcal{A}'_p))$ is regular and we can effectively construct a corresponding automaton $\mathcal{A}_p$ from $\mathcal{A}'_p$ and $\mathcal{A}'$. Similarly we can construct automata for the languages

$$L_A := \{\alpha\#^n \otimes w\#^k \mid \alpha \otimes w \in L(\mathcal{A}') \wedge \alpha\#^n \in P \wedge |\alpha\#^n| = |w\#^k|\} \text{ and}$$
$$L_R := \{\alpha\#^n \otimes w_1\#^{k_1} \otimes \ldots \otimes w_r\#^{k_r} \mid \alpha \otimes w_1 \otimes \ldots \otimes w_r \in L(\mathcal{A}'_R) \wedge \alpha\#^n \in P \wedge |\alpha\#^n| = |w_i\#^{k_i}|\}.$$

Together these automata form the presentation we are looking for. ◀

▶ **Lemma 25.** *Let $\mathcal{C}$ be a uniformly automatic class of* finite *structures. From a given automatic presentation $(P, \mathfrak{c})$ of $\mathcal{C}$ one can effectively construct a uniformly automatic presentation $(P^\times, \mathfrak{c}^\times)$ of $\mathcal{C}^\times$ and $(P^\uplus, \mathfrak{c}^\uplus)$ of $\mathcal{C}^\uplus$. Moreover, regularity of the advice set is preserved.*

Proof. Let $\mathfrak{c} = (\mathcal{A}, (\mathcal{A}_R)_{R\in\tau}, \pi)$ be an automatic presentation of $\mathcal{C}$ over the advice set P. Construction of $(P^\times, \mathfrak{c}^\times)$: By Lemma 30, we might assume that for all $\alpha \in P$ and all $w \in L(\mathcal{A}[\alpha])$ we have $|\alpha| = |w|$. As parameter set for $\mathcal{C}^\times$ we can now take $(P\#)^*P$, where $\alpha_1\# \ldots \#\alpha_n$ is an advice for $\mathcal{S}(\mathfrak{c}[\alpha_1]) \times \cdots \times \mathcal{S}(\mathfrak{c}[\alpha_n])$. The construction of a uniform presentation $\mathfrak{c}^\times$ of $\mathcal{C}^\times$ from $\mathfrak{c}$ is straight forward. On reading $\alpha_1\# \ldots \#\alpha_n$ as parameter,

the automaton $\mathcal{A}^\times$ should accept exactly the words $w_1\# \dots \# w_n$ with $w_i \in L(\mathcal{A}[\alpha_i])$ for all $i \in \{1, \dots, n\}$. This can obviously be done by an automaton since all words in $L(\mathcal{A}[\alpha_i])$ have the same length as α_i. The same holds for the presentation of the relations $R \in \tau$.

Construction of $(P^\times, \mathfrak{c}^\times)$: We construct a presentation over the advice set $(P\#)^*P$, where $\alpha_1\# \cdots \#\alpha_n$ should be an advice for $\biguplus_{1 \le i \le n} \mathcal{S}(\mathfrak{c}[\alpha_i])$. For an advice $\alpha_1\# \cdots \#\alpha_n$ we encode the elements of $\mathcal{S}(\alpha_i)$, $1 \le i \le n$, by the language $L_i = \#^{|\alpha_1\#\cdots\#\alpha_{i-1}\#|} L(\mathcal{A}[\alpha])$. Intuitively we shift the encodings of the elements in the copy of the i-th summand so that it matches again with the beginning of the i-th advice. Obviously one can construct a parameterised automaton with $L(\mathcal{A}[\alpha]) = \bigcup_{1 \le i \le n} L_i$ for all $\alpha = \alpha_1\# \cdots \#\alpha_n \in (P\#)^*P$. It is an easy exercise to construct the rest of the presentation. ◀

▶ **Theorem 29.** *The problem to decide whether a regularly automatic presentation has the unique representation property is Π^0_1-complete for the empty signature and for signatures with only monadic predicates (even for classes of finite structures), and Π^1_1-hard for any signature with at least one predicate of arity at least two.*

Proof. Π^1_1-hardness for signatures with binary relations follows directly from the the fact that the isomorphism problem for automatic structures is Σ^1_1-complete [15]. Obviously, given automatic presentations $\mathfrak{d}_0, \mathfrak{d}_1$, one can construct a parameterised presentation $\mathfrak{c}$ over the parameters $\{0, 1\}$ such that $\mathcal{S}(\mathfrak{c}[0]) = \mathcal{S}(\mathfrak{d}_0)$ and $\mathcal{S}(\mathfrak{c}[1]) = \mathcal{S}(\mathfrak{d}_1)$. Then $\mathfrak{c}$ has the unique representation property if, and only if, $\mathcal{S}(\mathfrak{d}_0) \ncong \mathcal{S}(\mathfrak{d}_1)$.

In order to establish Π^0_1-completeness for the empty signature we adopt a technique used by Kuske et al. in [18] to show that the ismomorphism problem for automatic equivalence relations is Π^0_1-complete. More precisely we use encodings of polynomials by automata to reduce Hilbert's 10th problem to the uniqueness problem. The problem can be formulated as follows: given polynomials $p, q \in \mathbb{N}[x_1, \dots, x_k]$ decide whether $p(\overline{a}) = q(\overline{a})$ for some $\overline{a} \in \mathbb{N}^k$. In [18] it is shown that for every polynomial p with nonegative coefficients one can construct an automaton $\mathcal{A}_p$ such that on input $1^{n_1} \otimes \dots \otimes 1^{n_k}$ the automaton $\mathcal{A}_p$ has exactly $p(n_1, \dots, n_k)$ accepting runs [18, Lemma 2]. Given an automaton $\mathcal{A}_p$ we can construct the following automatic presentation $\mathfrak{c}_p$ of the class $\{(\{0, \dots, m-1\}) \mid \exists \overline{n}(p(\overline{n}) = m)\}$. The parameter language of $\mathfrak{c}$ is $\{1^{n_1} \otimes \dots \otimes 1^{n_k} \mid \overline{n} \in \mathbb{N}^k\}$. For a parameter α the domain language is $\{w \in Q_p^* \mid w \text{ is an accepting run of } \mathcal{A}_p \text{ on } \alpha\}$, which is uniformly automatic since an automaton can check while reading $\alpha \otimes w$ if w is an accepting run of $\mathcal{A}_p$ on α. To complete the proof note that we can construct injective polynomials C_k for any arity k. For $p, q \in \mathbb{N}[x_1, \dots, x_n]$ define $p' := C_{k+1}(x_1, \dots, x_k, p(x_1, \dots, x_k))$ and $q' := C_{k+1}(x_1, \dots, x_k, q(x_1, \dots, x_k))$. Then p' and q' are both injective and $p'(\overline{a}) = q'(\overline{b})$ holds if, and only if, $\overline{a} = \overline{b}$ and $p(\overline{a}) = q(\overline{b})$. Now let $\mathfrak{c}$ be the parameter disjoint union of $\mathfrak{c}_{p'}$ and $\mathfrak{c}_{q'}$. By the aforementioned properties of p', q', $\mathfrak{c}$ has the unique representation property if, and only if. $p(\overline{a}) \neq q(\overline{a})$ for all $\overline{a} \in \mathbb{N}^k$. This establishes the hardness for Π^0_1. Further, the isomorphism problem is decidable for automatic structures with purely monadic signatures. Hence the uniqueness problem is in Π^0_1 since we can just enumerate all pairs of distinct parameters (α, β) from the regular parameter set and check if $\mathcal{S}(\mathfrak{c}[\alpha]) \cong \mathcal{S}(\mathfrak{c}[\beta])$. ◀

Strongly Normalizing Audited Computation*†

Wilmer Ricciotti[1] and James Cheney[2]

1 **LFCS, University of Edinburgh, Edinburgh, UK**
`wricciot@inf.ed.ac.uk`
2 **LFCS, University of Edinburgh, Edinburgh, UK**
`jcheney@inf.ed.ac.uk`

Abstract

Auditing is an increasingly important operation for computer programming, for example in security (e.g. to enable history-based access control) and to enable reproducibility and accountability (e.g. provenance in scientific programming). Most proposed auditing techniques are ad hoc or treat auditing as a second-class, extralinguistic operation; logical or semantic foundations for auditing are not yet well-established. *Justification Logic* (JL) offers one such foundation; Bavera and Bonelli introduced a computational interpretation of JL called λ^h that supports auditing. However, λ^h is technically complex and strong normalization was only established for special cases. In addition, we show that the equational theory of λ^h is inconsistent. We introduce a new calculus λ^{hc} that is simpler than λ^h, consistent, and strongly normalizing. Our proof of strong normalization is formalized in Nominal Isabelle.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Lambda calculus, Justification Logic, Strong Normalization, Audited Computation

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.36

1 Introduction

Auditing is of increasing importance in many areas, including in security (e.g. history-based access control [1], evidence-based audit [24, 15]) and to provide reproducibility or accountability for scientific computations (e.g. provenance tracking [3]). To date, most approaches to auditing propose to instrument programs in some *ad hoc* way, to provide a record of the computation, sometimes called *provenance*, *audit log*, *trace*, or *trail*, which is typically meant to be inspected after the computation finishes. Foundations of provenance or auditing have not been fully developed, and auditing typically remains a second-class citizen (or even an alien concept), in that trails are not accessible to the program itself during execution, but only to an external monitoring layer. We seek logical and semantic foundations for *self-auditing programs* that provide first-class recording and auditing primitives.

In this paper, we explore how ideas from *justification logic* [5] (a generalization of the *Logic of Proofs* [6]) can provide such a foundation. *Justification Logic* is the general name for a family of logics that refine the epistemic reading of modal necessity $\Box A$ ("A is known") by indexing such formulas with *justifications*, or (descriptions of) evidence that A holds. So, the formula $[\![a]\!]\, A$ can be read as "justification a is evidence (proof) establishing A". In elementary

* An extended version of this paper is available at `https://arxiv.org/abs/1706.03711`.
† Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-13-1-3006. The U.S. Government and University of Edinburgh are authorised to reproduce and distribute reprints for their purposes notwithstanding any copyright notation thereon.

forms of justification logic, the justifications consist of uninterpreted constants that can be combined with algebraic operations (e.g. formal sum and product). A representative axiom of justification logic is the evidence-aware version of Modus Ponens:

$$[\![a]\!]\,(A \supset B) \supset [\![b]\!]\,A \supset [\![a \cdot b]\!]\,B$$

which can be read as "if a proves $A \supset B$ and b proves A then a together with b proves B". Another representative axiom is the reflection principle:

$$[\![a]\!]\,A \supset [\![!a]\!]\,[\![a]\!]\,A$$

which can be read as "if a proves A then $!a$ proves that a proves A", where ! is a *reflection* operator on justifications. Notice that in both cases, if we ignore the evidence annotations a, b, we get standard axioms "K" and "4" of modal logic respectively [16].

This paper explores using (constructive) justification logic as a foundation for programming with auditing, building on of previous work on calculi for modal logic [21] and justification logic [7, 9]. Artemov and Bonelli [7] introduced the *intensional lambda calculus* $\lambda^{\mathbf{I}}$, which builds upon Pfenning and Davies' type theory for modal logic [21] by recording *proof codes* in judgements and modalities; proof codes are a special case of proof terms. In $\lambda^{\mathbf{I}}$, "trails" that witness the equivalence of proof codes were introduced, but these trails could not be inspected by other constructs. More recently, Bavera and Bonelli [9] introduced λ^h, an extension of $\lambda^{\mathbf{I}}$ that allowed *trail inspection*, which can audit a computation's trail of evaluation from the currently-enclosing box constructor up to the point of inspection.

Trail inspection is a first-class construct of λ^h, and can be performed at arbitrary program points (including inside another trail inspection). Bavera and Bonelli outlined how this construct can be used to implement *history-based access control* [1] policies in which access control decisions are made based on the complete previous call history. Trail inspection could also be used for profiling, for example by counting the number of calls of each function in a subcomputation, or for more general auditing [4], for example by comparing the trails of multiple computations to look for differences or anomalies in their execution history. Furthermore, all of these applications can be supported in a single language, rather than requiring ad hoc changes to accommodate different monitoring semantics. (It is worth noting that these applications typically use data structures such as sets or maps that are not provided in λ^h; however, they are straightforward to add.)

Bavera and Bonelli proved basic metatheoretic properties of λ^h including subject reduction, and gave a partial proof of strong normalization. The latter proof considered only terms in which there was no nesting of box constructors. Although they conjectured that strong normalization holds for the full system, the general case was left as an open question. However, λ^h is a somewhat complex system, due to subtle aspects of the metatheory of trail inspection. Trails are viewed as proof terms witnessing equivalence of proof codes. Proof codes include trail inspections, whose results depend on the history of the term being executed. λ^h's trails are viewed as equality proofs and are thus subject to symmetry: operationally, the language provides a trail constructor $\mathbf{s}(q)$ that converts a trail q of type $s = t$ to one of type $t = s$. Since the reduct of an outer trail inspection depends on a non-local, volatile context (the trail declared in the nearest outer box constructor) symmetry makes it possible to equate any two codes of the same type: in other words, the theory of trails becomes inconsistent.

To avoid this problem, Bavera and Bonelli annotated trail inspections with affine trail variables α, α' representing the volatile context, and trail inspection is then denoted by $\alpha\theta$, where θ is a collection of functions specifying a structural recursion over the trail. Nevertheless, in λ^h we can prove any two codes s and t equivalent. A detailed example, along with the rules of λ^h, are presented in Appendix A; here, we sketch the derivation:

$$\dfrac{\dfrac{s = q_1\theta \qquad q_1\theta = \alpha\theta}{s = \alpha\theta} \qquad \dfrac{\dfrac{t = q_2\theta \qquad q_2\theta = \alpha\theta}{t = \alpha\theta}}{\alpha\theta = t}}{s = t}\ (*)$$

where q_1, q_2 and θ are chosen so that $s = q_1\theta$ and $t = q_2\theta$, which is possible whenever s and t have the same type. The problem seems to originate from the transitivity rule marked with (*), which constitutes an exception to the affineness of trail variables, since the same proof code must necessarily appear as the right-hand side of the first premise and as the left-hand side of the second one. This allows trail inspection on α to be triggered twice. It should be noted that this property does *not* contradict any proved results by Bavera and Bonelli; instead, it can be viewed as an (undesirable) form of proof-irrelevance: that is, $[\![s]\!]\, A \to [\![t]\!]\, A$ is inhabited for any well-formed proof codes s, t. Of course, if this is the case then the stated justification s for $[\![s]\!]\, A$ need have nothing with to do its actual proof, defeating the purpose of introducing the proof codes in the first place.

We present a new calculus, λ^{hc}, that overcomes these issues with λ^h, and provides a more satisfactory foundation for audited computation, including the following contributions:

- We dispense with λ^h's trail variables, instead adopting a trail inspection construct $\iota(\theta)$ that can be used anywhere in the program. We also adjust λ^h's treatment of trails by no longer viewing them as witnesses to proof code *equality* but rather to (directed) *reducibility*. At a technical level, this simply consists of dropping the symmetry axiom and trail form. Viewing trails as asymmetric reducibility proofs means that it is no longer inconsistent to relate a trail inspection redex with all of its possible contracta nondeterministically, and it also obviates the need for trail variables (or ordinary variables) to be affine; once we lift these restrictions, however, the rationale for trail variables evaporates. We present the improved system, and summarize its metatheory (including a proof that its equational theory is consistent) in Section 2.
- To show that λ^{hc} is well-behaved, we prove strong normalization (SN). Our proof of SN actually shows a stronger result: SN holds even if we conservatively assume that trail inspection operations can choose *any* well-formed trail; this is a useful insight because it means that we can extend the result to more realistic languages with more complex trails. We summarize this proof, which we fully formalized in Nominal Isabelle[1].

Section 4 discusses related work: in particular, we argue that our approach can be used to prove strong normalization for λ^h as well, even in the presence of the symmetry rule.

2 A history-aware calculus

2.1 Overview

In this section we describe λ^{hc}, an extension of the simply-typed lambda calculus allowing for several forms of auditing, thanks to an internalized notion of *computation history*. It is based in many respects on λ^h, but since our preferred notation differs in important details we give a self-contained presentation.

Operationally, we add to the simply typed λ-calculus a new term form $!_q M$, where M is an expression and q is a *trail* containing a symbolic description of the past history of M. History-aware expressions will be given their own type: if M has type A, and q is an

[1] The full formalization can be found at `http://www.wilmer-ricciotti.net/FORMAL/ccau.tgz`. A more detailed discussion can be found in the extended version of this paper.

appropriate trail showing how M has reduced, then the type of $!_q M$ will be indicated by $[\![s]\!]\, A$; the symbol '!' is called "bang". In our language, we will initially evaluate expressions in the form $!\ M$, where the absence of a subscript indicates that no computation has happened within M yet. As we reduce M, the bang will be annotated with a subscript, for example

$$!\ M \rightsquigarrow !_q N$$

where q must explain how N was obtained from the initial expression M.

As a slightly more involved example, we consider the following expression:

$$!\ \mathsf{if}\ (1 \stackrel{?}{=} 2)\ M\ N$$

Here, $\stackrel{?}{=}$ represents an equality test between integers, while M and N represent respectively the 'then' and 'else' branches of the if statement. To evaluate the expression, we first reduce the equality test to $\#F$ (boolean false), and the if to its 'else' branch. In λ^{hc}, this computation will be described as:

$$!\ \mathsf{if}\ (1 \stackrel{?}{=} 2)\ M\ N \rightsquigarrow !_{\mathcal{Q}[q]}\mathsf{if}\ \#F\ M\ N \rightsquigarrow !_{\mathbf{t}(\mathcal{Q}[q],q')} N$$

where q, $\mathcal{Q}$, and q' are defined in such a way that:

- q explains how $1 \stackrel{?}{=} 2$ reduces to $\#F$
- $\mathcal{Q}$ is a context indicating that q was applied to the guard of an if statement – in other words, it describes a congruence rule of reduction
- q' describes the reduction of $\mathsf{if}\ \#F \cdots$ to its 'else' branch
- $\mathbf{t}$ combines its two arguments by means of transitivity.

To make use of history-aware expressions, we will also provide constructs to access their contents or to inspect their history.

2.2 Syntax

We now give the formal definition of λ^{hc}. The language includes three syntactic classes, which we have already presented informally in the overview:

- *codes* s, t correspond to (extended) lambda expressions which have not been evaluated yet: in other words, they only contain bangs with no history
- *trails* q, q' encode the history of the computation transforming a certain code into another
- *terms* M, N are lambda expressions that have been partially evaluated: they are similar to codes, but their bangs are annotated with trails.

The *types* of the language (denoted by metavariables $A, B, C, \ldots$) include atomic types P, functions $A \supset B$, and *audited units* $[\![s]\!]\, A$. Notice that types contain codes, which can contain types in turn. Audited types correspond to terms that capture and record all the computation happening inside them, for subsequent retrieval.

The syntax of the language is presented in Figure 1. Codes and terms are defined in a similar way to Pfenning and Davies [21]. We use the symbol '!' ("bang") to denote box introduction in both codes and terms; the corresponding elimination form is given by the let operator, which unpacks an audited code or term allowing us to access its contents. The variable declared by a let is bound in its second argument: in essence, $\mathrm{let}(u := !_q M, N)$ will reduce to N, where u has been replaced by M; q will be used to produce a trail explaining this reduction.

Codes:

$$
\begin{array}{lcll}
s,t,\ldots & ::= & a & \\
& | & u & \\
& | & \lambda a^A.s & \\
& | & (s\ t) & \\
& | & !s & \\
& | & \mathrm{let}(u^A := s, t) & \\
& | & \iota(\theta) &
\end{array}
$$

Terms:

$$
\begin{array}{lcll}
M,N,\ldots & ::= & a & \text{simple variable} \\
& | & u & \text{audited variable} \\
& | & \lambda a^A.M & \text{abstraction} \\
& | & (M\ N) & \text{application} \\
& | & !_q M & \text{box introduction} \\
& | & \mathrm{let}(u^A := M, N) & \text{box elimination} \\
& | & \iota(\vartheta) & \text{trail inspection}
\end{array}
$$

Trails:

$$
\begin{array}{lcll}
q,q',\ldots & ::= & \mathbf{r}(s) & \text{reflexivity} \\
& | & \mathbf{t}(q,q') & \text{transitivity} \\
& | & \beta(a^A.s,t) & \text{beta reduction} \\
& | & \beta_{\square}(s,u^A.t) & \text{box beta reduction} \\
& | & \mathbf{ti}(q,\theta) & \text{trail inspection reduction} \\
& | & \mathbf{lam}(a^A.q) & \text{congruence wrt lambda} \\
& | & \mathbf{app}(q,q') & \text{congruence wrt application} \\
& | & \mathbf{let}(q,u^A.q') & \text{congruence wrt let} \\
& | & \mathbf{trpl}(\zeta) & \text{congruence wrt trail insp.}
\end{array}
$$

Trail inspection branches:

$$
\begin{array}{lcll}
\theta,\theta',\ldots & ::= & \{\overrightarrow{s/\psi}\} & \text{for codes} \\
\vartheta,\vartheta',\ldots & ::= & \{\overrightarrow{M/\psi}\} & \text{for terms} \\
\zeta,\zeta',\ldots & ::= & \{\overrightarrow{q/\psi}\} & \text{for trails}
\end{array}
$$

Trail inspection branch labels:

$$
\begin{array}{lcl}
\psi,\psi',\ldots & ::= & \mathbf{r} \mid \mathbf{t} \mid \beta \mid \beta_{\square} \mid \mathbf{ti} \\
& | & \mathbf{lam} \mid \mathbf{app} \mid \mathbf{let} \\
& | & \mathbf{trpl}_{[]} \mid \mathbf{trpl}_{::} \mid \mathbf{d}
\end{array}
$$

Figure 1 Syntax of λ^{hc}.

▶ **Example 1.** λ^{hc} allows us to manipulate history-carrying data explicitly. For instance, we could have the following history-carrying natural numbers:

$$!_q 2 : [\![1+1]\!]\,\mathbb{N} \qquad !_{q'} 6 : [\![\mathsf{fact}\ 3]\!]\,\mathbb{N}$$

The trail q represents the history of 2, i.e. a witness to the computation that produced 2 by adding $1+1$; similarly, q' describes how computing $\mathsf{fact}\ 3$ (for a suitable definition of the factorial function) yielded 6.

Now supposing we wish to add these two numbers together, at the same time retaining their history, we will use the let construct to look inside them:

$$!\ \mathrm{let}(u^{\mathbb{N}} := !_q 2, \mathrm{let}(v^{\mathbb{N}} := !_{q'} 6, u+v)) \overset{*}{\rightsquigarrow} !_{q''} 8 : \left[\!\!\left[\mathrm{let}(u^{\mathbb{N}} := !\ 1+1, \mathrm{let}(v^{\mathbb{N}} := !\ \mathsf{fact}\ 3, u+v))\right]\!\!\right]\ \mathbb{N}$$

where the type of the terms is preserved under reduction, and the final trail q'', produced by composing q and q', is expected to log the reductions of $1+1$, of $\mathsf{fact}\ 3$, and of the two lets.

Trail inspection codes and terms will perform computation by primitive recursion on a certain trail, which represents an audited unit's computation history. Trail inspection references the computation history of the current context ($\iota(\theta)$ and $\iota(\vartheta)$). Here, θ and ϑ contain the branches that define the recursion.

The only difference between codes and terms is that in terms, bang operators are annotated with a trail q: this follows the intuition that terms are codes considered with their computational evolution. Trails represent (multi-step) reductions relating two codes: thus, their structure matches the derivation tree of the reduction itself.

In this definition, when a trail contains an argument composed of a variable and another expression separated by a dot, the variable should be considered bound in that expression. The constructors $\beta, \beta_{\square}$ and **ti** define atomic reductions: β represents the regular β-reduction; $\beta_{\square}$ is similar, but reduces a let-bang combination; finally, **ti** reduces trail inspections. The constructors **lam**, **app**, **let**, and **trpl** provide congruence rules for all code constructors (with

the notable exception of bangs, since trails are not allowed to escape an audited unit): for example, if q is a trail from s to t, $\mathbf{lam}(a^A.q)$ constructs a trail from $\lambda a^A.s$ to $\lambda a^A.t$. Finally, the definition of trails is completed by reflexivity and transitivity (but excluding symmetry). Notice that reflexive trails take as a parameter the code whose identity is stated (for brevity, we will keep this parameter implicit in our examples).

▶ **Example 2.** We can build a pair of natural numbers using Church's encoding: λ^{hc} makes it possible to keep a log of this operation without adding specialized code:

$$\begin{aligned} & !\ ((\lambda x, y, p.p\ x\ y)\ 2)\ 6 \\ \rightsquigarrow & !_{\mathbf{t}(\mathbf{r},\mathbf{app}(\boldsymbol{\beta}(x.\lambda y,p.p\ x\ y,2),\mathbf{r}))}\ (\lambda y, p.p\ 2\ y)\ 6 \\ \rightsquigarrow & !_{\mathbf{t}(\mathbf{t}(\mathbf{r},\mathbf{app}(\boldsymbol{\beta}(x.\lambda y,p.p\ x\ y,2),\mathbf{r})),\boldsymbol{\beta}(y.\lambda p.p\ 2\ y,6))}\ \lambda p.p\ 2\ 6 \end{aligned}$$

The trail for the first computation step is obtained by transitivity (trail costructor $\mathbf{t}$) from the original trivial trail ($\mathbf{r}$, i.e. reflexivity) composed with $\boldsymbol{\beta}(x.\lambda y, p.p\ x\ y, 2)$, which describes the beta-reduction of $(\lambda x, y, p.p\ x\ y)\ 2$: this subtrail is wrapped in a congruence **app** because the reduction takes place deep inside the left-hand subterm of an application (the other argument of **app** is reflexivity, because no reduction takes place in the right-hand subterm).

The second beta-reduction happens at the top level and is thus not wrapped in a congruence. It is combined with the previous trail by means of transitivity.

Inspection branches θ, ϑ and ζ are maps from trail constructors to codes, terms, or trails. More precisely, since some of the trail constructors have a variable arity, in that case we need to distinguish a nil case ($[]$) and a cons case ($::$) to specify the corresponding branch of recursion; in addition, we do not require the map to be exhaustive, but allow a "default", catch-all branch specified using the label $\mathbf{d}$: this does not make the language more expressive, but will allow us to extend the language with additional constants without needing to modify the base syntax. We now give a formal definition of the result of trail inspection.

▶ **Definition 3.** The operation $q\theta$, which produces a code by structural recursion on q applying the inspection branches θ, is defined as follows:

$$\begin{aligned} \psi(\_)\theta &\triangleq \theta(\mathbf{d}) \quad (\text{if } \psi \notin \mathrm{dom}(\theta)) & \mathbf{lam}(a^A.q)\theta &\triangleq \theta(\mathbf{lam})\ (q\theta) \\ \mathbf{r}(\_)\theta &\triangleq \theta(\mathbf{r}) & \mathbf{app}(q,q')\theta &\triangleq \theta(\mathbf{app})\ (q\theta)\ (q'\theta) \\ \mathbf{t}(q,q')\theta &\triangleq \theta(\mathbf{t})\ (q\theta)\ (q'\theta) & \mathbf{let}(q,u^A.q')\theta &\triangleq \theta(\mathbf{let})\ (q\theta)\ (q'\theta) \\ \boldsymbol{\beta}(\_,\_)\theta &\triangleq \theta(\boldsymbol{\beta}) & \mathbf{trpl}(\{\})\theta &\triangleq \theta(\mathbf{trpl}_{[]}) \\ \boldsymbol{\beta}_{\square}(\_,\_)\theta &\triangleq \theta(\boldsymbol{\beta}_{\square}) & \mathbf{trpl}(\{q/\psi, \overrightarrow{q'/\psi'}\})\theta &\triangleq \theta(\mathbf{trpl}_{::})\ (q\theta)\ (\mathbf{trpl}(\{\overrightarrow{q'/\psi'}\})\theta) \\ \mathbf{ti}(\_,\_)\theta &\triangleq \theta(\mathbf{ti}) \end{aligned}$$

The twin operation $q\vartheta$, which produces a term by structural recursion on q, is defined similarly, by replacing each occurrence of θ with ϑ.

▶ **Example 4.** For profiling purposes, we might be interested in counting the number of computation steps that have happened thus far in an audited unit. This can be accomplished by means of a trail replacement ϑ_+ such that:

$$\vartheta_+(\psi) = \begin{cases} 1 & \text{if } \psi = \boldsymbol{\beta}, \boldsymbol{\beta}_{\square}, \mathbf{ti} \\ \lambda x.x & \text{if } \psi = \mathbf{lam} \\ \lambda x, y.x + y & \text{if } \psi = \mathbf{t}, \mathbf{app}, \mathbf{let}, \mathbf{trpl}_{::} \\ 0 & \text{else} \end{cases}$$

This trail replacement counts 1 for each reduction step ($\boldsymbol{\beta}, \boldsymbol{\beta}_{\square}, \mathbf{ti}$); the cases for transitivity and congruences add together the recursive results for subtrails; the last clause ignores other constructors, most notably $\mathbf{r}$ and $\mathbf{ti}_{[]}$, which are not reduction steps and do not have subtrails.

2.3 Substitutions

λ^{hc} employs two notions of substitution, one for each kind of variable supported by the language:

- Simple variable substitution, which is not history-aware, is applied to codes, types, or terms and maps a simple variable to, respectively, a code or a term; we denote it by $[t/a]$ or $[N/a]$, which can also be written γ for short.
- Audited variable substitution, which is history-aware, is applied to codes or terms and maps an audited variable to, respectively, a code t or a tuple (N, q, t); we denote it by $[t/u]$ or $[(N, q, t)/u]$, which can also be written δ for short.

We will also use the letter η to refer to either of the two flavors of substitution when more precision is not needed.

▶ **Definition 5** (substitution on codes). Substitution of simple and audited variables on a code r ($r[s/a]$ and $r[s/u]$) is defined as usual (avoiding variable capture). The full definition is shown in Appendix B.1.

Each of the two substitutions on codes is extended to trails and types in the obvious way, by traversing the trail or type until a subexpression containing a code s is found, then resorting to substitution on s. To define substitution on terms, we first need auxiliary definitions of the *source* and *target* of a trail:

▶ **Definition 6.** The source and target of a trail q ($\mathrm{src}(q)$ and $\mathrm{tgt}(q)$) are defined as follows:

$$\begin{aligned}
\mathrm{src}(\mathbf{r}(s)) &\triangleq s & \mathrm{tgt}(\mathbf{r}(s)) &\triangleq s \\
\mathrm{src}(\mathbf{t}(q_1, q_2)) &\triangleq \mathrm{src}(q_1) & \mathrm{tgt}(\mathbf{t}(q_1, q_2)) &\triangleq \mathrm{tgt}(q_2) \\
\mathrm{src}(\boldsymbol{\beta}(a^A.s_1, s_2)) &\triangleq (\lambda a^A.s_1)\ s_2 & \mathrm{tgt}(\boldsymbol{\beta}(a^A.s_1, s_2)) &\triangleq s_1\,[s_2/a] \\
\mathrm{src}(\boldsymbol{\beta}_{\square}(s_1, v^A.s_2)) &\triangleq \mathrm{let}(v^A := !s_1, s_2) & \mathrm{tgt}(\boldsymbol{\beta}_{\square}(s_1, v^A.s_2)) &\triangleq s_2\,[s_1/v] \\
\mathrm{src}(\mathbf{ti}(q', \theta)) &\triangleq \iota(\theta) & \mathrm{tgt}(\mathbf{ti}(q', \theta)) &\triangleq q'\theta
\end{aligned}$$

The omitted cases are routine and shown in Appendix B.2.

A valid trail q represents evidence that $\mathrm{src}(q)$ reduces to $\mathrm{tgt}(q)$ (we formally state this result in the next section).

We omit the definition of simple variable substitution on terms for brevity (it is similar to the corresponding substitution on codes). The definition of audited variable substitution on terms, however, deserves some more explanation: since this substitution arises from the unpacking of a bang term, which contains a trail we need to take care of, it takes the form $[(N, q, t)/u]$, where q is the computation history that led from the code t to the term N.

For related reasons, this substitution can be applied to a term to produce both a new term and a corresponding trail: the two operations are written $M \times \delta$ and $M \ltimes \delta$, respectively.

▶ **Definition 7** (audited variable substitution (terms)). The operations $M \times [(N, q, t)/u]$ and $M \ltimes [(N, q, t)/u]$ are defined as follows:

$$\begin{aligned}
u \ltimes [(N, q, t)/u] &\triangleq N & (!_{q'} R) \ltimes [(N, q, t)/u] &\triangleq !_{\mathbf{t}(q'[t/u], R \times [(N,q,t)/u])}(R \ltimes [(N, q, t)/u]) \\
u \times [(N, q, t)/u] &\triangleq q & (!_{q'} R) \times [(N, q, t)/u] &\triangleq \mathbf{r}(!(\mathrm{src}(q')\,[t/u]))
\end{aligned}$$

The omitted cases are routine and shown in Appendix B.3. The main feature of interest here is that when we $\ltimes$-substitute (N, q, t) for u in an audited unit $!_{q'}$, we need to augment the trail with a step showing how the replaced occurrence of u evaluated from t to N, as well as substituting for other occurrences of u; the corresponding case of $\times$-substitution returns the appropriate reflexivity trail, since no trail escapes a bang as a result of substitution.

▶ **Example 8.** We consider again the term from Example 1:

$$!\ \mathrm{let}(u^{\mathbb{N}} := !_q 2, \mathrm{let}(v^{\mathbb{N}} := !_{q'} 6, u + v))$$

Audited variable substitution is used to reduce let-! combinations; for example, to reduce the outer let in the example, we need to compute

$$\mathrm{let}(v^{\mathbb{N}} := !_{q'} 6, u + v) \ltimes [2, q, \mathrm{src}(q)/u] = \mathrm{let}(v^{\mathbb{N}} := !_{q'} 6, 2 + v)$$
$$\mathrm{let}(v^{\mathbb{N}} := !_{q'} 6, u + v) \times [2, q, \mathrm{src}(q)/u] = \mathbf{let}(\mathbf{r}, \mathbf{app}(\mathbf{app}(\mathbf{r}, q), \mathbf{r}))$$

(we assume that the infix notation for + is syntactic sugar for two nested applications).

2.4 Type system

We now introduce typing rules for codes, trails, and terms. These rules use three corresponding judgments with the following shape:

$\Delta; \Gamma \vdash s : A$	s has type A
$\Delta; \Gamma \vdash q : s \stackrel{\triangleright}{=}_A t$	q witnesses s reducing to t, with type A
$\Delta; \Gamma \vdash M : A \mid s$	M has type A and code s

The environments Δ and Γ are list of type declarations for audited and simple variables respectively, in the form $u :: A$ or $a : A$.

Figure 2 shows the typing rules of the language. In the rules for codes, if we forget for a moment about trail inspections, it is easy to recognize Pfenning and Davies's judgmental presentation of modal logic. The rules for terms are very similar, the biggest difference being found in rule T-Bang: to typecheck $!_q M$, we first obtain the type A and code t for M; then we typecheck the trail q to ensure that it has type A and target t; if both checks are successful, we can return $[\![s]\!]\, A$ as the type of the expression, and $!s$ as its code, where s is the source of q. This also shows that when we say that $!s$ is the code of $!_q M$, we mean that the term is the result of reducing the code, and the trail q records the computation history.

The rules for trail inspection, both as a code and as a term, rely on an auxiliary definition of $\mathcal{T}^B$, which is a function from inspection branch labels to the corresponding type; it depends on the superscript B, which refers to the output type of the inspection. The rule for inspection codes (TI) checks that the types of all the branches in θ match their labels. This requires θ to be defined on the default branch $\mathbf{d}$ to guarantee that the inspection is exhaustively defined. We define $\mathcal{T}^B$ as follows:

$$\mathcal{T}^B(\psi) \triangleq \begin{cases} B & (\psi = \mathbf{d}, \mathbf{r}, \beta, \beta_{\square}, \mathbf{ti}, \mathbf{iti}, \mathbf{trpl}_{[]}) \\ B \supset B & (\psi = \mathbf{lam}, \mathbf{itb}_{[]}) \\ B \supset B \supset B & (\psi = \mathbf{t}, \mathbf{app}, \mathbf{let}, \mathbf{itb}_{::}, \mathbf{trpl}_{::}) \end{cases}$$

Rather than typecheck a term to compute its code, we can define a function performing this operation directly.

▶ **Definition 9.** The code of a term M (notation: $\mathrm{cd}(M)$) is the code obtained by replacing all occurrences of $!_q N$ with $!\mathrm{src}(q)$. The full definition is shown in Appendix B.4.

$\boxed{\Delta;\Gamma \vdash s : A}$

$$\frac{a : A \in \Gamma}{\Delta;\Gamma \vdash a : A}\ \textsc{Var} \qquad \frac{\Delta;\Gamma, a : A \vdash s : B}{\Delta;\Gamma \vdash \lambda a^A.s : A \supset B}\ \supset I \qquad \frac{\Delta;\Gamma \vdash s : A \supset B \qquad \Delta;\Gamma \vdash t : A}{\Delta;\Gamma \vdash s\ t : B}\ \supset E$$

$$\frac{u :: A \in \Delta}{\Delta;\Gamma \vdash u : A}\ \textsc{mVar} \qquad \frac{\Delta;\cdot \vdash t : A}{\Delta;\Gamma \vdash !t : [\![t]\!]\ A}\ \Box I \qquad \frac{\Delta;\Gamma \vdash s : [\![r]\!]\ A \qquad \Delta, u :: A;\Gamma \vdash t : C}{\Delta;\Gamma \vdash \mathrm{let}(u^A := s, t) : C\,[r/u]}\ \Box E$$

$$\frac{\mathbf{d} \in \mathrm{dom}(\theta) \qquad \left[\Delta;\cdot \vdash \theta(\psi) : \mathcal{T}^B(\psi)\right]_{\psi \in \mathrm{dom}(\theta)}}{\Delta;\Gamma \vdash \iota(\theta) : B}\ \textsc{TI}$$

$\boxed{\Delta;\Gamma \vdash M : A \mid s}$

$$\frac{a : A \in \Gamma}{\Delta;\Gamma \vdash a : A \mid a}\ \textsc{T-Var} \qquad \frac{\Delta;\Gamma, a : A \vdash M : B \mid s}{\Delta;\Gamma \vdash \lambda a^A.M : A \supset B \mid \lambda a^A.s}\ \textsc{T-Abs}$$

$$\frac{\Delta;\Gamma \vdash M : A \supset B \mid s \qquad \Delta;\Gamma \vdash N : A \mid t}{\Delta;\Gamma \vdash M\ N : B \mid s\ t}\ \textsc{T-App}$$

$$\frac{u :: A \in \Delta}{\Delta;\Gamma \vdash u : A \mid u}\ \textsc{T-mVar} \qquad \frac{\Delta;\cdot \vdash M : A \mid t \qquad \Delta;\cdot \vdash q : s \stackrel{\triangleright}{=}_A t}{\Delta;\Gamma \vdash !_q M : [\![s]\!]\ A \mid !s}\ \textsc{T-Bang}$$

$$\frac{\Delta;\Gamma \vdash M : [\![r]\!]\ A \mid s \qquad \Delta, u :: A;\Gamma \vdash N : C \mid t}{\Delta;\Gamma \vdash \mathrm{let}(u^A := M, N) : C\,[r/u] \mid \mathrm{let}(u^A := s, t)}\ \textsc{T-Let}$$

$$\frac{\mathbf{d} \in \mathrm{dom}(\vartheta) \qquad \mathrm{dom}(\vartheta) = \mathrm{dom}(\theta) \qquad \left[\Delta;\cdot \vdash \vartheta(\psi) : \mathcal{T}^B(\psi) \mid \theta(\psi)\right]_{\psi \in \mathrm{dom}(\vartheta)}}{\Delta;\Gamma \vdash \iota(\vartheta) : B \mid \iota(\theta)}\ \textsc{T-TI}$$

$\boxed{\Delta;\Gamma \vdash q : s \stackrel{\triangleright}{=}_A t}$

$$\frac{\Delta;\Gamma \vdash s : A}{\Delta;\Gamma \vdash \mathbf{r}(s) : s \stackrel{\triangleright}{=}_A s}\ \textsc{Q-Refl} \qquad \frac{\Delta;\Gamma \vdash q_1 : r \stackrel{\triangleright}{=}_A s \qquad \Delta;\Gamma \vdash q_2 : s \stackrel{\triangleright}{=}_A t}{\Delta;\Gamma \vdash \mathbf{t}(q_1, q_2) : r \stackrel{\triangleright}{=}_A t}\ \textsc{Q-Trans}$$

$$\frac{\Delta;\Gamma, a : A \vdash s : B \qquad \Delta;\Gamma \vdash t : A}{\Delta;\Gamma \vdash \beta(a^A.s, t) : (\lambda a^A.s)\ t \stackrel{\triangleright}{=}_B s\,[t/a]}\ \textsc{Q-}\beta$$

$$\frac{\Delta;\cdot \vdash s : A \qquad \Delta, u :: A;\Gamma \vdash t : C}{\Delta;\Gamma \vdash \beta_{\Box}(s, u^A.t) : \mathrm{let}(u^A := !s, t) \stackrel{\triangleright}{=}_{C[s/u]} t\,[s/u]}\ \textsc{Q-}\beta_{\Box}$$

$$\frac{\mathbf{d} \in \mathrm{dom}(\vartheta) \qquad \Delta;\cdot \vdash q : s \stackrel{\triangleright}{=}_A t \qquad \left[\Delta;\cdot \vdash \theta(\psi) : \mathcal{T}^B(\psi)\right]_{\psi \in \mathrm{dom}(\theta)}}{\Delta;\Gamma \vdash \mathbf{ti}(q, \theta) : \iota(\theta) \stackrel{\triangleright}{=}_B q\theta}\ \textsc{Q-TI}$$

$$\frac{\Delta;\Gamma, a : A \vdash q : s \stackrel{\triangleright}{=}_B t}{\Delta;\Gamma \vdash \mathbf{lam}(a^A.q) : \lambda a^A.s \stackrel{\triangleright}{=}_{A \supset B} \lambda a^A.t}\ \textsc{Q-Abs}$$

$$\frac{\Delta;\Gamma \vdash q_1 : s_1 \stackrel{\triangleright}{=}_{A \supset B} t_1 \qquad \Delta;\Gamma \vdash q_2 : s_2 \stackrel{\triangleright}{=}_A t_2}{\Delta;\Gamma \vdash \mathbf{app}(q_1, q_2) : s_1\ s_2 \stackrel{\triangleright}{=}_B t_1\ t_2}\ \textsc{Q-App}$$

$$\frac{\Delta;\Gamma \vdash q_1 : s_1 \stackrel{\triangleright}{=}_{[\![r]\!]A} t_1 \qquad \Delta, u :: A;\Gamma \vdash q_2 : s_2 \stackrel{\triangleright}{=}_C t_2}{\Delta;\Gamma \vdash \mathbf{let}(q_1, u^A.q_2) : \mathrm{let}(u^A := s_1, s_2) \stackrel{\triangleright}{=}_{C[r/u]} \mathrm{let}(u^A := t_1, t_2)}\ \textsc{Q-Let}$$

$$\frac{\mathbf{d} \in \mathrm{dom}(\zeta) \qquad \mathrm{dom}(\zeta) = \mathrm{dom}(\theta) = \mathrm{dom}(\theta') \qquad \left[\Delta;\cdot \vdash \zeta(\psi) : \theta(\psi) \stackrel{\triangleright}{=}_{\mathcal{T}^B(\psi)} \theta'(\psi)\right]_{\psi \in \mathrm{dom}(\zeta)}}{\Delta;\Gamma \vdash \mathbf{trpl}(\zeta) : \iota(\theta) \stackrel{\triangleright}{=}_B \iota(\theta')}\ \textsc{Q-Trpl}$$

Figure 2 Typing rules for λ^{hc}.

The rule T-Bang, which typechecks audited terms, needs to compute the type of a trail: this is performed by the typechecking rules for trails. Each of these rules typechecks a different trail constructor: in particular, four rules define the contraction of redexes (Q-$\boldsymbol{\beta}$, Q-$\boldsymbol{\beta}_{\blacksquare}$, Q-TI), Q-Refl and Q-Trans ensure that trails induce a preorder on codes, and the remaining rules model congruence with respect to all code constructors but bangs. Let us remark that the trail q mentioned in trail inspections cannot be synthesized from the redex (since codes are not history-aware), but must be provided as an argument to **ti**.

We can prove that operations of Definitions 6 and 9 match the typing judgments:

▶ **Lemma 10.** *If* $\Delta;\Gamma \vdash q : s \stackrel{\triangleright}{=}_A t$ *then* $\mathrm{src}(q) = s$ *and* $\mathrm{tgt}(q) = t$.

▶ **Lemma 11.** *If* $\Delta;\Gamma \vdash M : A \mid s$ *then* $\mathrm{cd}(M) = s$.

2.5 Semantics

In this section, we define a reduction relation expressing how terms compute to values. Reduction differs from trails since it relates terms (rather than codes), and since unlike trails it does not appear as part of terms (however, it is defined in such a way that reduced terms will contain modified trails expressing a *log* of reduction).

Our definition of reduction makes use of contexts, i.e. terms with holes, represented by black boxes (■); similarly, we provide a notion of trail contexts, denoted by $\mathcal{Q}$. The notation $\mathcal{E}[M]$ indicates the term obtained by filling the hole in $\mathcal{E}$ with M; $\mathcal{Q}[q]$ is defined similarly.

$$\begin{array}{rcl} \mathcal{E} & ::= & \blacksquare \mid \lambda a^A.\mathcal{E} \mid (\mathcal{E}\ M) \mid (M\ \mathcal{E}) \mid !_q\mathcal{E} \mid \mathrm{let}(u^A := \mathcal{E}, M) \mid \mathrm{let}(u^A := M, \mathcal{E}) \mid \iota(\{\overrightarrow{M/\psi}, \mathcal{E}/\psi', \overrightarrow{N/\psi''}\}) \\ \mathcal{Q} & ::= & \blacksquare \mid \mathbf{t}(\mathcal{Q}, q) \mid \mathbf{t}(q, \mathcal{Q}) \mid \mathbf{app}(\mathcal{Q}, q) \mid \mathbf{app}(q, \mathcal{Q}) \mid \mathbf{lam}(a^A.\mathcal{Q}) \mid \mathbf{let}(\mathcal{Q}, u^A.q) \mid \mathbf{let}(q, u^A.\mathcal{Q}) \\ & \mid & \mathbf{trpl}(\{\overrightarrow{q_1/\psi}, \mathcal{Q}/\psi', \overrightarrow{q_2/\psi''}\}) \end{array}$$

Following [9] we use $\mathcal{F}$ to refer to contexts that do not allow holes inside bangs. Box-free contexts $\mathcal{F}$ and trail contexts $\mathcal{Q}$ are related by the following definition:

▶ **Definition 12** (canonical trail context)**.** For every $\mathcal{F}$, there exists a canonical trail context $\mathcal{Q}_{\mathcal{F}}$, defined as follows:

$$\begin{array}{rclrcl} \mathcal{Q}_{\blacksquare} & \triangleq & \blacksquare & \mathcal{Q}_{(\mathcal{F}\ M)} & \triangleq & \mathbf{app}(\mathcal{Q}_{\mathcal{F}}, \mathbf{r}(\mathrm{cd}(M))) \\ \mathcal{Q}_{(M\ \mathcal{F})} & \triangleq & \mathbf{app}(\mathbf{r}(\mathrm{cd}(M)), \mathcal{Q}_{\mathcal{F}}) & \mathcal{Q}_{\lambda a^A.\mathcal{F}} & \triangleq & \mathbf{lam}(a^A.\mathcal{Q}_{\mathcal{F}}) \\ \mathcal{Q}_{\mathrm{let}(\mathcal{F}, u^A.M)} & \triangleq & \mathbf{let}(\mathcal{Q}_{\mathcal{F}}, u.\mathbf{r}(\mathrm{cd}(M))) & \mathcal{Q}_{\mathrm{let}(M, u^A.\mathcal{F})} & \triangleq & \mathbf{let}(\mathbf{r}(\mathrm{cd}(M)), u^A.\mathcal{Q}_{\mathcal{F}}) \\ \multicolumn{6}{c}{\mathcal{Q}_{\iota(\{\overrightarrow{M/\psi}, \mathcal{F}/\psi', \overrightarrow{N/\psi''}\})} \triangleq \mathbf{trpl}(\{\overrightarrow{\mathbf{r}(\mathrm{cd}(M))/\psi}, \mathcal{Q}_{\mathcal{F}}/\psi'.\overrightarrow{\mathbf{r}(\mathrm{cd}(N))/\psi''}\})} \end{array}$$

Informally, $\mathcal{Q}_{\mathcal{F}}$ uses congruences to express a reflexive trail for $\mathcal{F}$, with a hole as the subtrail corresponding to the hole in $\mathcal{F}$. It is $\mathcal{Q}_{\mathcal{F}}$ that is responsible for producing the **app** congruence of Example 2.

Thanks to trail contexts and our definition of audited variable substitution, we can define reduction directly, without an auxiliary judgment pushing trails towards the closest outer bang. To avoid dealing with the unwanted situation of trail inspections not guarded by a bang, we only consider terms surrounded by an outer bang.

The reduction rules are defined in Figure 3. They operate by contracting a subterm appearing in a context $\mathcal{F}$, at the same time updating the trail of the enclosing bang by asserting that q is followed by a new contraction appearing in the trail context $\mathcal{Q}_{\mathcal{F}}$. Rule B-$\beta_{\square}$ is an exception, in that after performing the $\boldsymbol{\beta}_{\blacksquare}$ contraction, we still need to take into account the residuals of q', expressed by the substitution $N \times [(M, q', \mathrm{src}(q'))/u]$

We write $s \rightsquigarrow t$ when there exists a well-typed q such that $\mathrm{src}(q) = s$ and $\mathrm{tgt}(q) = t$.

$$\frac{}{!_q\mathcal{F}[(\lambda a^A.M)\ N)] \rightsquigarrow !_{\mathbf{t}(q,\mathcal{Q}_\mathcal{F}[\beta(a.\mathrm{cd}(M),\mathrm{cd}(N))])}\mathcal{F}[M\,[N/a]]}\ \text{B-}\beta \qquad \frac{M \rightsquigarrow N}{!_q\mathcal{F}[M] \rightsquigarrow !_q\mathcal{F}[N]}\ \text{B-BANG}$$

$$\frac{q_f = \mathbf{t}(q, \mathcal{Q}_\mathcal{F}[\mathbf{t}(\beta_\square(\mathrm{src}(q'), \mathrm{cd}(N)), N \times [(M, q', \mathrm{src}(q'))/u])])}{!_q\mathcal{F}[\mathrm{let}(!_{q'}M, u.N)] \rightsquigarrow !_{q_f}\mathcal{F}[N \ltimes [(M, q', \mathrm{src}(q'))/u]]}\ \text{B-}\beta_\square$$

$$\frac{}{!_q\mathcal{F}[\iota(\vartheta)] \rightsquigarrow !_{\mathbf{t}(q,\mathcal{Q}_\mathcal{F}[\mathbf{ti}(q,\mathrm{cd}(\vartheta))])}\mathcal{F}[q\vartheta]}\ \text{B-TI}$$

Figure 3 Term reduction rules for λ^{hc}.

▶ **Example 13.** We take again the term from Example 1 and reduce the outer let as follows:

$$!\ \mathrm{let}(u^{\mathbb{N}} := !_q 2, \mathrm{let}(v^{\mathbb{N}} := !_{q'} 6, u + v)) \rightsquigarrow !_{q_f}\ \ \mathrm{let}(v^{\mathbb{N}} := !_{q'} 6, 2 + v)$$

where we use the substitutions we computed in Example 8. Since the reduction happens immediately inside the bang, we use rule B-$\beta_\square$ with $\mathcal{Q}_\blacksquare = \blacksquare$, thus q_f is as follows:

$$\begin{aligned} q_f &= \mathbf{t}(\mathbf{r}, \mathbf{t}(\beta_\square(\mathrm{src}(q), u.\mathrm{cd}(\mathrm{let}(v := !_{q'} 6, u + v)), \mathbf{let}(\mathbf{r}, \mathbf{app}(\mathbf{app}(\mathbf{r}, q), \mathbf{r})))) \\ &= \mathbf{t}(\mathbf{r}, \mathbf{t}(\beta_\square(1 + 1, u.\,\mathrm{let}(v := !\ \mathsf{fact}\ 3, u + v)), \mathbf{let}(\mathbf{r}, \mathbf{app}(\mathbf{app}(\mathbf{r}, q), \mathbf{r})))) \end{aligned}$$

(where src and cd compute according to the hypotheses we made about q and q').

▶ **Example 14.** To show how to use the profiling inspection from Example 4, we need to assume a certain evaluation strategy: for example, call-by-value. We then write $(x \leftarrow M; N)$ as syntactic sugar for $(\lambda x.N)\ M$ and evaluate the following term:

$$\begin{aligned} &!\ (t_0 \leftarrow \iota(\vartheta_+); \\ &\quad \_ \leftarrow ((\lambda x, y, p.p\ x\ y)\ 2)\ 6; \\ &\quad t_1 \leftarrow \iota(\vartheta_+); \\ &\quad t_1 - t_0) \end{aligned}$$

In this term, t_0 evaluates to 0 because the bang starts with a reflexive trail; by the time we get to the third line, the outer trail contains a log of the inspection on the first line, the two beta reductions needed to evaluate the second line, and two more beta reductions to account for sequential compositions: thus t_1 evaluates to 5; finally, we evaluate $t_1 - t_0 = 5$.

▶ **Definition 15** (normal form)**.** A term M is in normal form iff for all trails q and all terms N, $!_q M \not\rightsquigarrow N$. Likewise, a code is in normal form iff there exists no $t \neq s$ such that $s \rightsquigarrow t$.

It should be noted that λ^{hc}, despite being strongly normalizing (as we will prove in the next section), is not confluent: the same term may reduce to different values under different reduction strategies. In particular, the trails appearing in bangs are sensitive to the reduction order: a call-by-value strategy and a call-by-name strategy will produce different trails. We could recover confluence by forcing a certain evaluation strategy and quotienting trails by means of a suitable equivalence relation: this is beyond the scope of the present paper.

The main properties of λ^{hc}, such as subject reduction, can be proved similarly to those for λ^h. As explained in Section 1, we also need to establish the consistency of trails as proofs of reducibility.

▶ **Theorem 16.** *For all codes s, t of λ^{hc} in normal form such that $s \neq t$, there exist no Δ, Γ, q, A such that $\Delta; \Gamma \vdash q : s \overset{\triangleright}{=}_A t$*

Proof. If such a judgment were provable, by structural induction on it we would be able to show that q must be a combination of reflexivity, transitivity, and congruence rules (since both s and t are in normal form, no contraction is possible in absence of symmetry). Then $s = t$, which falsifies the hypothesis. ◀

3 Strong Normalization

We now summarize our proof of strong normalization for λ^{hc}. The presence of a recursion operator on trails, whose occurrences can be arbitrarily nested, together with the fact that trails grow monotonically during execution, makes this result non-trivial.

Our proof employs the well-known notion of "candidates of reducibility" [13] (sets of strongly normalizing terms enjoying certain desirable properties). Candidates of reducibility are a powerful and flexible tool, which has been used to prove the strong normalization property of very expressive lambda calculi [12, 18, 25] and also other results such as the Church-Rosser property [11].

In the literature, it is possible to find several definitions of candidate of reducibility: along with Girard's definition, we can cite Tait's saturated sets [23] and Parigot's inductive definition [19]. Our proof employs Girard's candidates, and can be easily compared to the similar proof for the System F [14]. Some acquaintance with that proof will be assumed in the rest of this section. We will use $\mathcal{SN}$ to denote the set of all strongly normalizing terms. Reduction on strongly normalizing terms is a well-founded relation, which allows us to reason by well-founded induction on it.

3.1 A simplified calculus

As a technical means to investigate normalization of λ^{hc}, we define λ^{hs}, a simplified version of the calculus which forgets about trails associated with box introductions. Its types, terms, and contexts are defined by the following grammar:

$$\begin{array}{rcl} \tau, \sigma & ::= & P \mid \tau \supset \sigma \mid \Box\tau \\ s, t, \ldots & ::= & a \mid u \mid \lambda a^{\tau}.s \mid (s\ t) \mid !s \mid \mathrm{let}(u^{\tau} := s, t) \mid \iota(\theta) \\ \theta, \theta', \ldots & ::= & \{\overrightarrow{s/\psi}\} \\ \mathcal{E} & ::= & \blacksquare \mid \lambda a^{\tau}.\mathcal{E} \mid (\mathcal{E}\ s) \mid (s\ \mathcal{E}) \mid !\mathcal{E} \mid \mathrm{let}(u^{\tau} := \mathcal{E}, s) \mid \mathrm{let}(u^{\tau} := s, \mathcal{E}) \mid \iota(\{\overrightarrow{s/\psi}, \mathcal{E}/\psi', \overrightarrow{t/\psi''}\}) \end{array}$$

The terms of λ^{hs} correspond closely to the codes of λ^{hc}. Their semantics, however, is different: $\iota(\theta)$ does not perform inspection on the trail of the enclosing bang, but at the time of its evaluation will receive an arbitrary trail. We omit the definition of trails for brevity, but it can be obtained from the corresponding notion in λ^{hc}, by replacing codes with λ^{hs} terms.

We can also define simple and audited variable substitution on λ^{hs} terms, in a way that mimics the corresponding notion of λ^{hc} (Definition 5).

We provide an erasure map from λ^{hc} types and terms to λ^{hs} (its extension to contexts is immediate).

$$\begin{array}{ccccc} |P| = P & & |A \supset B| = |A| \supset |B| & & |[\![s]\!]\, A| = \Box|A| \\ |a| \triangleq a & |u| \triangleq u & |\lambda a^{A}.M| \triangleq \lambda a^{|A|}.|M| & & |M\ N| \triangleq |M|\ |N| \\ |!_q M| \triangleq !|M| & & |\,\mathrm{let}(u^{A} := M, N)| \triangleq \mathrm{let}(u^{A} := |M|, |N|) & & |\iota(\vartheta)| \triangleq \iota(|\vartheta|) \end{array}$$

The typing rules and reduction rules for λ^{hs} are given in Figures 4 and 5. They are similar to their counterparts in λ^{hc}, but greatly simplified: in particular, trail inspection is allowed to reduce by nondeterministically picking any trail.

Unsurprisingly, erasure preserves well-typedness and reduction:

$$\frac{a : \tau \in \Gamma}{\Delta;\Gamma \vdash a : \tau} \qquad \frac{u :: \tau \in \Delta}{\Delta;\Gamma \vdash u : \tau} \qquad \frac{\Delta;\Gamma, a : \tau \vdash M : \sigma}{\Delta;\Gamma \vdash \lambda a^{\tau}.M : \tau \supset \sigma} \qquad \frac{\Delta;\cdot \vdash M : \tau}{\Delta;\Gamma \vdash !M : \Box\tau}$$

$$\frac{\Delta;\Gamma \vdash M : \tau \supset \sigma \quad \Delta;\Gamma \vdash N : \tau}{\Delta;\Gamma \vdash M\ N : \sigma} \qquad \frac{\Delta;\Gamma \vdash M : \Box\tau \quad \Delta, u :: \tau;\Gamma \vdash N : \sigma}{\Delta;\Gamma \vdash \mathrm{let}(u^{\tau} := M, N) : \sigma} \qquad \frac{\mathbf{d} \in \mathrm{dom}(\theta) \quad [\Delta;\cdot \vdash \theta(\psi) : \mathcal{T}^{\sigma}(\psi)]_{\psi \in \mathrm{dom}(\theta)}}{\Delta;\Gamma \vdash \iota(\theta) : \sigma}$$

Figure 4 Typing rules for λ^{hs} terms.

$$((\lambda a.s)\ t) \rightsquigarrow s\,[t/a] \qquad \mathrm{let}(!s, u.t) \rightsquigarrow s\,[t/u] \qquad \iota(\theta) \rightsquigarrow q\theta \qquad \frac{s \rightsquigarrow t}{\mathcal{E}[s] \rightsquigarrow \mathcal{E}[t]}$$

Figure 5 Reduction rules for λ^{hs} terms.

▶ **Lemma 17.** *If* $\Delta;\Gamma \vdash_{\lambda^{hc}} M : A | s$*, then* $\Delta;\Gamma \vdash_{\lambda^{hs}} |M| : |A|$.

▶ **Lemma 18.** $M \rightsquigarrow_{\lambda^{hc}} N \Longrightarrow |M| \rightsquigarrow_{\lambda^{hs}} |N|$

By the combination of Lemma 17 and Lemma 18, we know that if λ^{hs} is strongly normalizing, then λ^{hc} must also, *a fortiori*, be strongly normalizing. We will thus proceed to prove SN in the simpler system, and extend it to λ^{hc} as a corollary.

3.2 Summary of the proof

We now give the definition of candidates of reducibility *à la Girard*.

▶ **Definition 19** (neutral term)**.** A term is neutral if it is not of the form $\lambda a^{A}.s$ or $!s$.

▶ **Definition 20** (candidates of reducibility)**.** A set $\mathcal{C}$ of terms is a *candidate of reducibility* iff it satisfies Girard's CR conditions:
CR1. $\mathcal{C} \subseteq \mathcal{SN}$
CR2. $s \in \mathcal{C} \wedge s \rightsquigarrow t \Longrightarrow t \in \mathcal{C}$
CR3. $s \in \mathcal{NT} \wedge (\forall t. s \rightsquigarrow t \Longrightarrow t \in \mathcal{C}) \Longrightarrow s \in \mathcal{C}$.
The set of candidates of reducibility will be denoted $\mathcal{CR}$.

We identify certain subsets of candidates that are stable wrt. validity substitution:

▶ **Definition 21** (substitutive sub-candidate)**.** For all candidates $\mathcal{C}$, validity variables u, and sets of terms $\mathcal{D}$, we define its substitutive subset $\mathcal{C}_u^{\mathcal{D}}$ as $\mathcal{C}_u^{\mathcal{D}} \triangleq \{s \in \mathcal{C} : \forall t \in \mathcal{D}, s\,[t/u] \in \mathcal{C}\}$.

Notice that sub-candidates are not in $\mathcal{CR}$; they do, however, satisfy CR1 and CR2, which is enough for us to use them in the definition of *reducible sets*.

▶ **Definition 22** (reducible set)**.** For all types τ, the set Red_{τ} of reducible terms of type τ is defined by recursion on τ as follows:

- $\mathsf{Red}_P = \mathcal{SN}$
- $\mathsf{Red}_{\tau\supset\sigma} = \{s : \forall t \in \mathsf{Red}_{\tau}, (s\ t) \in \mathsf{Red}_{\sigma}\}$
- $\mathsf{Red}_{\Box\tau} = \{s : \forall u, \forall \mathcal{C} \in \mathcal{CR}, \forall t \in \mathcal{C}_u^{\mathsf{Red}_{\tau}}, \mathrm{let}(u := s, t) \in \mathcal{C}\}$

In particular, $s \in \mathsf{Red}_{\Box\tau}$ if, and only if, for all reducibility candidates $\mathcal{C}$ and audited variables u, if we take a term $t \in \mathcal{C}_u^{\mathsf{Red}_{\tau}}$, then $\mathrm{let}(u := s, t) \in \mathcal{C}$. We are allowed to use Red_{τ} in $\mathcal{C}_u^{\mathsf{Red}_{\tau}}$ because τ is a subexpression of $\Box\tau$.

▶ **Lemma 23.** *For each type* τ*,* $\mathsf{Red}_{\tau} \in \mathcal{CR}$.

▶ **Theorem 24.** *Let* $\Delta, \Gamma, \overrightarrow{\eta}$ *s.t.* $\mathrm{dom}(\overrightarrow{\eta}) = \mathrm{dom}(\Delta) \cup \mathrm{dom}(\Gamma)$*, for all* $u \in \mathrm{dom}(\Delta)$, $\overrightarrow{\eta}(u) \in \mathsf{Red}_{\Delta(u)}$*, and for all* $a \in \mathrm{dom}(\Gamma)$, $\overrightarrow{\eta}(a) \in \mathsf{Red}_{\Gamma(a)}$*. Then,* $\Delta; \Gamma \vdash s : \tau$ *implies* $s\overrightarrow{\eta} \in \mathsf{Red}_\tau$*.*

Proof. We use the standard technique for System F [14], with additional subproofs for $s \in \mathsf{Red}_\tau \implies !s \in \mathsf{Red}_{\Box\tau}$ and $(\forall \psi \in \mathrm{dom}(\theta), \theta(\psi) \in \mathsf{Red}_{\mathcal{T}^\sigma(\psi)}) \implies \iota(\theta) \in \mathsf{Red}_\sigma$. SN follows as a corollary when $\overrightarrow{\eta}$ is the identity substitution. ◀

3.3 Formalization

We formalized λ^{hc}, together with our proof of strong normalization, in Nominal Isabelle. Nominal Isabelle mechanizes the management of variable binding, relieving the user from the burden of defining binding infrastructure (e.g. de Bruijn indices, lifting operations, etc.). On the other hand, many of the definitions used in a nominal formalization must be proved to be "well-behaved", beyond what would usually be made explicit in a pencil-and-paper proof. The main well-behavedness property required in Nominal Isabelle is *equivariance*, stating that a function f or a set $\mathcal{S}$ is stable under finite permutations of names π:

$$\forall x. f(\pi \cdot x) = \pi \cdot f(x) \qquad \forall x. x \in \mathcal{S} \iff \pi \cdot x \in \mathcal{S}$$

Most of the definitions used in the proof are equivariant, including typing and reduction rules, the set $\mathcal{SN}$ of strongly normalizing terms, the set $\mathcal{CR}$ of reducibility candidates, reducibility sets Red_A for all types A, and the operator $\mathcal{C}^{\mathcal{D}}_u$.

We do not, however, prove equivariance for individual reducibility candidates $\mathcal{C} \in \mathcal{CR}$: on the contrary, reducibility candidates do not need to be equivariant. To prove it, one can take a closure operator $[\cdot]$ mapping a set of strongly normalizing terms to the smallest reducibility candidate containing it (for its existence and definition see e.g. [22]): it is easy to see that, for all variables a, b, $!a \in [\{!b\}]$ if and only if $a = b$.

Our impredicative definition of $\mathsf{Red}_{\Box\tau}$, which quantifies over arbitrary candidates, is handled gracefully by Nominal Isabelle. Lindley and Stark [17] show a technique ($\top\top$-lifting) that can be adapted to provide a predicative definition of $\mathsf{Red}_{\Box A}$. As it happens, the lower logical complexity of $\top\top$-lifting relies on the additional definition of *continuations*, which would require some more effort to be formalized. On the other hand, our approach seems likely to extend to handle structural recursion (System T), following Aschieri and Zorzi [8], or impredicative polymorphism (System F), following Girard et al. [14].

4 Related work

The first Justification Logic-style system, known as the Logic of Proofs, was introduced by Artemov [6, 5]. Most work on justification logic systems (as for modal logic) is presented via Hilbert-style axiom systems extending classical propositional logic. Pfenning and Davies' judgemental reconstruction of modal logic [21] provides a natural deduction-style proof system for (intuitionistic) modal logic with necessity and possibility modalities. Artemov and Bonelli [7] introduced a system $\lambda^{\mathbf{I}}$ for justification logic, extending Pfenning and Davies' treatment of necessity ($\Box A$). They introduced equality proofs (trails) to recover subject reduction and proved strong normalization for $\lambda^{\mathbf{I}}$. Bavera and Bonelli later introduced (outer) trail inspection as part of an extended calculus called λ^h [9] from which we took inspiration.

Our system λ^{hc} seems (at least to us) an improvement over λ^h: it is simpler, and avoids the inconsistency arising from viewing trails as equivalence proofs. Nevertheless, the technique we used to prove strong normalization in λ^{hc} seems to suffice for λ^h as well. The only complication concerns the definition of reduction: while in λ^{hc} reduction acts non-locally by

updating the trail in the nearest enclosing bang, in λ^h reduction produces an intermediate term annotated with a new local trail, and the trail in the enclosing bang is updated after a sequence of *permutation reductions*:

$$!_q\mathcal{F}[M] \rightsquigarrow !_q\mathcal{F}[q' \triangleright M'] \overset{*}{\rightsquigarrow} !_{q''}\mathcal{F}[M']$$

It is thus necessary, though not overly complicated, to redefine reduction as a single-step operation including permutation reductions, and prove its equivalence to the original definition. This allows us to remove intermediate terms $q \triangleright M$ from the calculus altogether. In general, the (efficient) reduction theory of systems such as λ^h and λ^{hc} deserves further study.

Our work is also partly motivated by previous work on provenance and tracing for functional languages. Perera et al. [20] presented a pure, ML-like core language in which program execution yields both a value and a *trace*, corresponding roughly to the large-step operational derivation leading to the result. They gave trace slicing algorithms and techniques for extracting program slices and differential slices from traces; in subsequent work Acar et al. [3] explored security implications such as the disclosure and obfuscation properties of traces [10]. Indeed, trail inspection can be considered as a generic mechanism for defining *provenance views* as considered by these papers. Although Bavera and Bonelli [9] motivated λ^h as a basis for history-based access control (following Abadi and Fournet [1]), we are not aware of comparable work on provenance security based on justification logic. Our ongoing investigation suggests that λ^h-style trails contain enough information to extract many forms of provenance; however, to perform this extraction by means of trail inspection, we would usually need to reverse beta-reductions, and in particular to undo the substitution in beta-reduced terms. Since inspections treat beta and beta-box trails as black boxes, this cannot be achieved in the current version of the calculus. An extension of the language providing transparent beta trails and operations to undo substitutions is the subject of our current study.

Audit has been considered by a number of security researchers recently, for example Amir-Mohamedian et al. [4] consider correctness for audit logging, but auditing is again an extralinguistic operation (implemented using aspect-oriented programming). Vaughan et al. [24] introduced new formalisms for evidence-based audit. In Aura, an implementation of this approach [15], dependently-typed programs execute in the presence of a policy specified in authorization logic [2], and whenever a restricted resource is requested, a proof that access is authorized is constructed at run time and stored in an audit log for later inspection. The relationship between this approach and ours, and more generally between justification logic and authorization logic, remains to be investigated.

5 Conclusions

The motivation for this work is the need to provide better foundations for audited computation, as advocated in recent work on provenance and security and on type-theoretic forms of justification logic such as λ^h. However, as we have shown, λ^h is at the same time overly restrictive (in requiring affine variable and trail variable usage, i.e. forbidding copying of ordinary data) and overly permissive: despite these restrictions aimed at keeping the equational theory consistent, one can still prove any two compatibly-typed proof codes equivalent, using symmetry and the nondeterministic equational law for trail inspection.

We presented a new calculus λ^{hc} based on justification logic that includes audit operations such as trail inspection, but has fewer limitations and is simpler than λ^h. We show that λ^h avoids this problem and has a consistent reduction theory. We also prove strong normalization

for λ^{hc} via a simplified system λ^{hs}, and we have mechanically checked the proof. This approach also seems sufficient to prove SN for λ^{h}, though we have not formalized this result.

In future work, we intend to consider larger-scale programming languages based on the ideas of λ^{hc}, investigate connections to provenance tracking and slicing techniques, and clarify the security guarantees offered by justification logic-based audit.

References

1 M. Abadi and C. Fournet. Access control based on execution history. In *NDSS*, 2003.

2 Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4), 1993. doi:10.1145/155183.155225.

3 U. A. Acar, A. Ahmed, J. Cheney, and R. Perera. A core calculus for provenance. *Journal of Computer Security*, 21:919–969, 2013. doi:10.1007/978-3-642-28641-4_22.

4 S. Amir-Mohammadian, S. Chong, and C. Skalka. Correct audit logging: Theory and practice. In *POST*, pages 139–162, 2016. doi:10.1007/978-3-662-49635-0_8.

5 S. Artemov. The logic of justification. *Review of Symbolic Logic*, 1(4):477–513, 2008. doi:10.1017/S1755020308090060.

6 S. N. Artëmov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, 2001. doi:10.2307/2687821.

7 S. N. Artëmov and E. Bonelli. The intensional lambda calculus. In *Logical Foundations of Computer Science*, pages 12–25, 2007. doi:10.1007/978-3-540-72734-7_2.

8 Federico Aschieri and Margherita Zorzi. Non-determinism, non-termination and the strong normalization of System T. In *TLCA*, pages 31–47, 2013. doi:10.1007/978-3-642-38946-7_5.

9 F. Bavera and E. Bonelli. Justification logic and audited computation. *Journal of Logic and Computation*, 2015. Published online, June 19, 2015. doi:10.1093/logcom/exv037.

10 J. Cheney. A formal framework for provenance security. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF)*, pages 281–293. IEEE, 2011. doi:10.1109/CSF.2011.26.

11 J. H. Gallier. On Girard's "candidats de réductibilité". In *Logic and Computer Science*, pages 123–203. Academic Press, 1990.

12 H. Geuvers. A short and flexible proof of strong normalization for the calculus of constructions. In *Selected Papers from TYPES'94*, LNCS, pages 14–38. Springer-Verlag, 1995. doi:10.1007/3-540-60579-7_2.

13 J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *Scandinavian Logic Symposium, 1978*, pages 63–92. North-Holland, 1971.

14 J.-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and Types*. Cambridge University Press, New York, NY, USA, 1989.

15 L. Jia, J. A. Vaughan, K. Mazurak, J. Zhao, L. Zarko, J. Schorr, and S. Zdancewic. Aura: a programming language for authorization and audit. In *ICFP*, pages 27–38, 2008. doi:10.1145/1411204.1411212.

16 C. I. Lewis. *Symbolic Logic*. Dover Publications, 1959.

17 Sam Lindley and Ian Stark. Reducibility and $\top\top$-lifting for computation types. In *TLCA*, number 3461 in LNCS, pages 262–277. Springer-Verlag, 2005. doi:10.1007/11417170_20.

18 Z. Luo. *An Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, 1990.

19 M. Parigot. Strong normalization for the second order classical natural deduction. In *Proceedings of LICS*, 1994.

20 R. Perera, U. A. Acar, J. Cheney, and P. B. Levy. Functional programs that explain their work. In *ICFP*, pages 365–376. ACM, 2012. doi:10.1145/2398856.2364579.

21 F. Pfenning and R. Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001. doi:10.1017/S0960129501003322.

22 Colin Riba. On the stability by union of reducibility candidates. In *FOSSACS*, pages 317–331, Berlin, Heidelberg, 2007. Springer-Verlag. doi:10.1007/978-3-540-71389-0_23.

23 W. W. Tait. A realizability interpretation of the theory of species. In *Logic Colloquium, 1972–73*, volume 453 of *Lecture Notes in Mathematics*, pages 240–251. Springer, 1975. doi:10.1007/BFb0064875.

24 J. A. Vaughan, L. Jia, K. Mazurak, and S. Zdancewic. Evidence-based audit. In *CSF*, pages 177–191, 2008. doi:10.1109/CSF.2008.24.

25 B. Werner. *Une Théorie des Constructions Inductives*. PhD thesis, Université Paris VII, 1994.

A The type system of λ^h

In Figures 6, 7, and 8, we provide the typing rules of Bavera and Bonelli's λ^h for reference, using syntax similar to that used in this paper.

A.1 Inconsistency of the equational theory of λ^h

In Section 1, we provided an informal derivation tree showing that the equational theory of λ^h allows to equate any two proof codes of the same type. We give below a more accurate (but isomorphic) derivation tree. Then, if we choose a θ such that $\theta(\mathbf{r}) = s$ and $\theta(\boldsymbol{\beta}) = t$, this effectively proves $s = t$, where s and t can be chosen freely:

$$\dfrac{\dfrac{\dfrac{\cdot;\cdot;\cdot \vdash \mathbf{r}(s') : s' =_B s'}{\cdot;\cdot;\cdot \vdash \theta : \mathcal{T}^A}\ \textsc{EqTI}}{\cdot;\cdot;\alpha : \mathsf{Eq}(A) \vdash \mathbf{ti}(\theta,\alpha) : \mathbf{r}(s')\theta =_A \alpha\theta}\ \textsc{EqTI} \qquad \dfrac{\dfrac{\cdot;\cdot;\cdot \vdash \boldsymbol{\beta}(b^{B'}.t', t'') : t'\left[t''/b\right] =_{B''} (\lambda b^{B'}.t')\ t''}{\cdot;\cdot;\cdot \vdash \theta : \mathcal{T}^A}\ \textsc{EqTI}}{\cdot;\cdot;\alpha : \mathsf{Eq}(A) \vdash \mathbf{ti}(\theta,\alpha) : \boldsymbol{\beta}(b^{B'}.t', t'')\theta =_A \alpha\theta}\ \textsc{EqSym}}{\cdot;\cdot;\alpha : \mathsf{Eq}(A) \vdash \mathbf{t}(\mathbf{ti}(\theta,\alpha), \mathbf{s}(\mathbf{ti}(\theta,\alpha))) : \mathbf{r}(s')\theta =_A \boldsymbol{\beta}(b^{B'}.t', t'')\theta}\ \textsc{EqTrans}$$

This derivation contains a lot of extraneous details, so we summarize the main features which contribute to the final result. In this derivation, we use transitivity on two subderivations. The right-hand subderivation shows that $s = \mathbf{r}(s')\theta$ is equal to $\alpha\theta$, using the rule EqTI. Here, s' is an arbitrary well-formed term of some type B. The EqTI rule allows us to conclude that any code of the form $q\theta$ is equal to $\alpha\theta$, and here we have used $q = \mathbf{r}(s')$.

In the second subderivation we use rule EqTI again to conclude that $t = \boldsymbol{\beta}(b^{B'}.t', t'')\theta$ is equivalent to $\alpha\theta$. Here, terms t' ad t'' are arbitrary well-formed terms of appropriate types to form a redex $\lambda b.t')\ t''$. We then use symmetry to show that $\alpha\theta$ is equivalent to $\boldsymbol{\beta}(b^{B'}.t', t'')\theta$. The desired conclusion follows. Since we can easily choose θ so that it returns one proof code s when handling a reflexivity trail and a different proof code t when handling a β-redex trail, the above derivation can be modified to prove that any two proof codes (of the same type A) are equivalent.

Finally, in applying the transitivity rule EqTrans, it is important that this rule does *not* require affine use of the trail variable α, so that it can be used in both subderivations. Changing this might fix the above problem, but would also have unknown consequences on the rest of the system. We have instead focused on an alternative system that does not require trail variables or affineness, by removing the symmetry trail.

$$\frac{a : A \in \Gamma}{\Delta;\Gamma;\Sigma \vdash a : A}\ \textsc{Var} \qquad \frac{u : A[\Sigma] \in \Delta \qquad \Sigma\sigma \subseteq \Sigma'}{\Delta;\Gamma;\Sigma' \vdash \langle u,\sigma\rangle : A}\ \textsc{mVar}$$

$$\frac{\Delta;\Gamma,a : A;\Sigma \vdash s : B}{\Delta;\Gamma \vdash \lambda a^A.s : A \supset B} \supset I \qquad \frac{\Delta;\Gamma_1;\Sigma_1 \vdash s : A \supset B \qquad \Delta;\Gamma_2;\Sigma_2 \vdash t : A}{\Delta;\Gamma_1,\Gamma_2;\Sigma_1,\Sigma_2 \vdash s\ t : B} \supset E$$

$$\frac{\Delta;\cdot;\Sigma \vdash s : A \qquad \Delta;\cdot;\Sigma \vdash q : s =_A t}{\Delta;\Gamma;\Sigma' \vdash \Sigma.t : [\![\Sigma.t]\!]\ A} \Box I \qquad \frac{\Delta;\Gamma_1;\Sigma_1 \vdash s : [\![\Sigma.r]\!]\ A \qquad \Delta,u : A[\Sigma];\Gamma_2;\Sigma_2 \vdash t : C}{\Delta;\Gamma_1,\Gamma_2;\Sigma_1,\Sigma_2 \vdash \mathrm{let}(u^{A[\Sigma]} := s,t) : C\,[\Sigma.r/u]} \Box E$$

$$\frac{\alpha : \mathsf{Eq}(A) \in \Sigma \qquad \Delta;\cdot;\cdot \vdash \theta : \mathcal{T}^B}{\Delta;\Gamma;\Sigma \vdash \alpha\theta : B}\ \textsc{TI} \qquad \frac{\Delta;\Gamma;\Sigma \vdash s : A \qquad \Delta;\Gamma;\Sigma \vdash q : s =_A t}{\Delta;\Gamma;\Sigma \vdash t : A}\ \textsc{Eq}$$

Figure 6 Typing rules for λ^h proof codes.

$$\frac{\Delta;\Gamma;\Sigma \vdash s : A}{\Delta;\Gamma;\Sigma \vdash \mathbf{r}(s) : s =_A s}\ \textsc{EqRefl} \qquad \frac{\Delta;\Gamma_1,a : A;\Sigma_1 \vdash s : A \supset B \qquad \Delta;\Gamma_2;\Sigma_2 \vdash t : A}{\Delta;\Gamma_1,\Gamma_2;\Sigma_1,\Sigma_2 \vdash \beta(a^A.s,t) : s\,[t/a] =_B (\lambda a^A.s)\ t}\ \textsc{Eq}\beta$$

$$\frac{\Delta;\cdot;\Sigma_1 \vdash A|r \qquad \Delta,u : A[\Sigma_1];\Gamma_2;\Sigma_2 \vdash C|t \qquad \Delta;\cdot;\Sigma_1 \vdash q : r =_A s \qquad \Gamma_2 \subseteq \Gamma_3 \quad \Sigma_2 \subseteq \Sigma_3}{\Delta;\Gamma_3;\Sigma_3 \vdash \beta_{\Box}(\Sigma_1.s,u^{A[\Sigma_1]}.t) : t\,[\Sigma_1.s/u] =_{C[\Sigma_1.s/u]} \mathrm{let}(u^{A[\Sigma_1]} := s,t)}\ \textsc{Eq}\beta_{\Box}$$

$$\frac{\Delta;\cdot;\Sigma_1 \vdash q : s =_A t \qquad \Delta;\cdot;\cdot \vdash \mathcal{T}^B|\theta \qquad \alpha : \mathsf{Eq}(A) \in \Sigma_2}{\Delta;\Gamma;\Sigma_2 \vdash \mathbf{ti}(\vartheta,\alpha) : q\theta =_B \alpha\theta}\ \textsc{EqTI}$$

$$\frac{\Delta;\Gamma;\Sigma \vdash q : s =_A t}{\Delta;\Gamma;\Sigma \vdash \mathbf{s}(q) : t =_A s}\ \textsc{EqSym} \qquad \frac{\Delta;\Gamma;\Sigma \vdash q_1 : r =_A s \qquad \Delta;\Gamma;\Sigma \vdash q_2 : s =_A t}{\Delta;\Gamma;\Sigma \vdash \mathbf{t}(q_1,q_2) : r =_A t}\ \textsc{EqTrans}$$

$$\frac{\Delta;\Gamma,a : A;\Sigma \vdash q : s =_B t}{\Delta;\Gamma;\Sigma \vdash \mathbf{lam}(a^A.q) : \lambda a^A.s =_{A\supset B} \lambda a^A.t}\ \textsc{EqAbs}$$

$$\frac{\Delta;\Gamma_1;\Sigma_1 \vdash q_1 : s_1 =_{A\supset B} t_1 \qquad \Delta;\Gamma_2;\Sigma_2 \vdash q_2 : s_2 =_A t_2}{\Delta;\Gamma_1,\Gamma_2;\Sigma_1,\Sigma_2 \vdash \mathbf{app}(q_1,q_2) : s_1\ s_2 =_B t_1\ t_2}\ \textsc{EqApp}$$

$$\frac{\Delta;\Gamma_1;\Sigma_1 \vdash q_1 : s_1 =_{[\![\Sigma_r]\!]A} t_1 \qquad \Delta,u : A[\Sigma];\Gamma_2;\Sigma_2 \vdash q_2 : s_2 =_C t_2}{\Delta;\Gamma_1,\Gamma_2;\Sigma_1,\Sigma_2 \vdash \mathbf{let}(q_1,u^{A[\Sigma]}.q_2) : \mathrm{let}(u^{A[\Sigma]} := s_1,s_2) =_{C[\Sigma.r/u]} \mathrm{let}(u^{A[\Sigma]} := t_1,t_2)}\ \textsc{EqLet}$$

$$\frac{\alpha : \mathsf{Eq}(A) \in \Sigma \qquad \Delta;\cdot;\cdot \vdash \overrightarrow{q} : \theta =_{\mathcal{T}^B} \theta'}{\Delta;\Gamma;\Sigma \vdash \mathbf{trpl}(\alpha,\overrightarrow{q}) : \alpha\theta =_B \alpha\theta'}\ \textsc{EqTrpl}$$

Figure 7 Typing rules for λ^h trails.

$$\frac{a : A \in \Gamma}{\Delta;\Gamma;\Sigma \vdash a : A \mid a}\ \textsc{TVar} \qquad \frac{u : A[\Sigma] \in \Delta \qquad \Sigma\sigma \subseteq \Sigma'}{\Delta;\cdot;\Sigma' \vdash \langle u,\sigma\rangle : A \mid \langle u,\sigma\rangle}\ \textsc{TmVar}$$

$$\frac{\Delta;\Gamma,a : A;\Sigma \vdash M : B \mid s}{\Delta;\Gamma;\Sigma \vdash \lambda a^A.M : A \supset B \mid \lambda a^A.s}\ \textsc{TAbs} \qquad \frac{\Delta;\Gamma_1;\Sigma_1 \vdash M : A \supset B \mid s \qquad \Delta;\Gamma_2;\Sigma_2 \vdash N : A \mid t}{\Delta;\Gamma_1,\Gamma_2;\Sigma_1,\Sigma_2 \vdash M\ N : B \mid s\ t}\ \textsc{TApp}$$

$$\frac{\Delta;\cdot;\Sigma \vdash M : A \mid s \qquad \Delta;\cdot;\Sigma \vdash q : s = t}{\Delta;\Gamma;\Sigma' \vdash !^{\Sigma}_{q} M : [\![\Sigma.t]\!]\ A \mid \Sigma.t}\ \textsc{TBox}$$

$$\frac{\Delta;\Gamma_1;\Sigma_1 \vdash M : [\![\Sigma.r]\!]\ A \mid s \qquad \Delta,u : A[\Sigma];\Gamma_2;\Sigma_2 \vdash N : C \mid t}{\Delta;\Gamma_1,\Gamma_2;\Sigma_1,\Sigma_2 \vdash \mathrm{let}(u^{A[\Sigma]} := M,N) : C\,[\Sigma.r/u] \mid \mathrm{let}(u^{A[\Sigma]} := s,t)}\ \textsc{TLet}$$

$$\frac{\alpha : \mathsf{Eq}(A) \in \Sigma \qquad \Delta;\cdot;\cdot \vdash \vartheta : \mathcal{T}^B \mid \theta}{\Delta;\Gamma;\Sigma \vdash \alpha\vartheta : B \mid \alpha\theta}\ \textsc{TTI} \qquad \frac{\Delta;\Gamma;\Sigma \vdash M : A \mid s \qquad \Delta;\Gamma;\Sigma \vdash q : s =_A t}{\Delta;\Gamma;\Sigma \vdash q \triangleright M : A \mid t}\ \textsc{TEq}$$

Figure 8 Typing rules for λ^h terms.

B Full definitions

B.1 Substitution on codes

For $\gamma = [t/a]$ and $\delta = [t/u]$, we define substitution of simple and audited variables on a code r (notation: $r\gamma$ and $r\delta$) as follows:

$$
\begin{aligned}
b\gamma &\triangleq \gamma(b) \\
v\gamma &\triangleq v \\
(\lambda b^A.s)\gamma &\triangleq \lambda b^A.s\gamma && (b\#a,t) \\
(s\ s')\gamma &\triangleq (s\gamma)\ (s'\gamma) \\
(!s)\gamma &\triangleq !s \\
\mathrm{let}(v^A := s, s')\gamma &\triangleq \mathrm{let}(v^A := s\gamma, s'\gamma) && (v\#t) \\
\iota(\theta)\gamma &\triangleq \iota(\theta\gamma)
\end{aligned}
$$

$$
\begin{aligned}
b\delta &\triangleq b \\
v\delta &\triangleq \delta(v) \\
(\lambda b^A.s)\delta &\triangleq \lambda b^A.s\delta && (b\#t) \\
(s\ s')\delta &\triangleq (s\delta)\ (s'\delta) \\
(!s)\delta &\triangleq !(s\delta) \\
\mathrm{let}(v^A := s, s')\delta &\triangleq \mathrm{let}(v^A := s\delta, s'\delta) && (v\#u,t) \\
\iota(\theta)\delta &\triangleq \iota(\theta\delta)
\end{aligned}
$$

where $\theta\gamma$ and $\theta\delta$ are defined pointwise and the notation $\gamma(b)$ is defined as t if $a = b$, and as b otherwise ($\delta(v)$ is defined similarly).

B.2 Source and target of a trail

The source and target of a trail q (notation: $\mathrm{src}(q)$ and $\mathrm{tgt}(q)$) are defined as follows:

$$
\begin{aligned}
\mathrm{src}(\mathbf{r}(s)) &\triangleq s \\
\mathrm{src}(\mathbf{t}(q_1,q_2)) &\triangleq \mathrm{src}(q_1) \\
\mathrm{src}(\boldsymbol{\beta}(a^A.s_1,s_2)) &\triangleq (\lambda a^A.s_1)\ s_2 \\
\mathrm{src}(\boldsymbol{\beta}_{\square}(s_1,v^A.s_2)) &\triangleq \mathrm{let}(v^A := !s_1, s_2) \\
\mathrm{src}(\mathbf{ti}(q',\theta)) &\triangleq \iota(\theta) \\
\mathrm{src}(\mathbf{lam}(a^A.q')) &\triangleq \lambda a^A.\mathrm{src}(q') \\
\mathrm{src}(\mathbf{app}(q',q'')) &\triangleq (\mathrm{src}(q')\ \mathrm{src}(q'')) \\
\mathrm{src}(\mathbf{let}(q',v^A.q'')) &\triangleq \mathrm{let}(v^A := \mathrm{src}(q'), \mathrm{src}(q'')) \\
\mathrm{src}(\mathbf{trpl}(\zeta)) &\triangleq \iota(\mathrm{src}(\zeta))
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{tgt}(\mathbf{r}(s)) &\triangleq s \\
\mathrm{tgt}(\mathbf{t}(q_1, q_2)) &\triangleq \mathrm{tgt}(q_2) \\
\mathrm{tgt}(\boldsymbol{\beta}(a^A.s_1, s_2)) &\triangleq s_1\,[s_2/a] \\
\mathrm{tgt}(\boldsymbol{\beta}_{\square}(s_1, v^A.s_2)) &\triangleq s_2\,[s_1/v] \\
\mathrm{tgt}(\mathbf{ti}(q', \theta)) &\triangleq q'\theta \\
\mathrm{tgt}(\mathbf{lam}(a^A.q')) &\triangleq \lambda a^A.\mathrm{tgt}(q') \\
\mathrm{tgt}(\mathbf{app}(q', q'')) &\triangleq (\mathrm{tgt}(q')\ \mathrm{tgt}(q'')) \\
\mathrm{tgt}(\mathbf{let}(q', v^A.q'')) &\triangleq \mathrm{let}(v^A := \mathrm{tgt}(q'), \mathrm{tgt}(q'')) \\
\mathrm{tgt}(\mathbf{trpl}(\zeta)) &\triangleq \iota(\mathrm{tgt}(\zeta))
\end{aligned}
$$

where $\mathrm{src}(\zeta)$ and $\mathrm{tgt}(\zeta)$ are defined pointwise.

B.3 Audited variable substitution

Given $\delta = [(N, q, t)/u]$, fix $\delta' = [t/u]$. Then, the operations $M \times \delta$ and $M \ltimes \delta$ are defined as follows:

$$
\begin{aligned}
a \times \delta &\triangleq \mathbf{r}(a) \\
v \times \delta &\triangleq \begin{cases} q & (u = v) \\ \mathbf{r}(v) & (u \neq v) \end{cases} \\
(\lambda a^A.R) \times \delta &\triangleq \mathbf{lam}(a^A.R \times \delta) && (\heartsuit) \\
(R\ S) \times \delta &\triangleq \mathbf{app}(R \times \delta, S \times \delta) \\
(!_{q'} R) \times \delta &\triangleq \mathbf{r}(!(\mathrm{src}(q')\delta')) \\
\mathrm{let}(v^A := R, S) \times \delta &\triangleq \mathbf{let}(R \times \delta, v^A.S \times \delta) && (\spadesuit) \\
\iota(\vartheta) \times \delta &\triangleq \mathbf{trpl}(\vartheta \times \delta)
\end{aligned}
$$

$$
\begin{aligned}
a \ltimes \delta &\triangleq a \\
v \ltimes \delta &\triangleq \begin{cases} N & (u = v) \\ v & (u \neq v) \end{cases} \\
(\lambda a^A.R) \ltimes \delta &\triangleq \lambda a^A.R \ltimes \delta && (\heartsuit) \\
(R\ S) \ltimes \delta &\triangleq (R \ltimes \delta)\ (S \ltimes \delta) \\
(!_{q'} R) \ltimes \delta &\triangleq !_{\mathbf{t}(q'\delta', R \times \delta)}(R \ltimes \delta) \\
\mathrm{let}(v^A := R, S) \ltimes \delta &\triangleq \mathrm{let}(v^A := R \ltimes \delta, u.S \ltimes \delta) && (\spadesuit) \\
\iota(\vartheta) \ltimes \delta &\triangleq \iota(\vartheta \ltimes \delta)
\end{aligned}
$$

$(\heartsuit)\quad a \# N, q, t \quad (\spadesuit)\quad v \# u, N, q, t$

where $\vartheta \times \delta$ and $\vartheta \ltimes \delta$ are defined pointwise on ϑ.

B.4 Codes

The code of a term M (notation: $\mathrm{cd}(M)$) is defined as follows:

$$\begin{aligned}
\mathrm{cd}(a) &\triangleq a \\
\mathrm{cd}(u) &\triangleq u \\
\mathrm{cd}(\lambda a^A.N) &\triangleq \lambda a^a.\mathrm{cd}(N) \\
\mathrm{cd}(N\ R) &\triangleq (\mathrm{cd}(N)\ \mathrm{cd}(R)) \\
\mathrm{cd}(!_q N) &\triangleq !\mathrm{src}(q) \\
\mathrm{cd}(\mathrm{let}(u^A := N, R)) &\triangleq \mathrm{let}(u^A := \mathrm{cd}(N), \mathrm{cd}(R)) \\
\mathrm{cd}(\iota(\vartheta)) &\triangleq \iota(\mathrm{cd}(\vartheta))
\end{aligned}$$

where $\mathrm{cd}(\vartheta)$ is defined pointwise.

A Finitary Analogue of the Downward Löwenheim-Skolem Property

Abhisekh Sankaran

Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Bombay, India
abhisekh@cse.iitb.ac.in

Abstract

We present a model-theoretic property of finite structures, that can be seen to be a finitary analogue of the well-studied downward Löwenheim-Skolem property from classical model theory. We call this property the $\mathcal{L}$-*equivalent bounded substructure property*, denoted $\mathcal{L}$-EBSP, where $\mathcal{L}$ is either FO or MSO. Intuitively, $\mathcal{L}$-EBSP states that a large finite structure contains a small "logically similar" substructure, where logical similarity means indistinguishability with respect to sentences of $\mathcal{L}$ having a given quantifier nesting depth. It turns out that this simply stated property is enjoyed by a variety of classes of interest in computer science: examples include regular languages of words, trees (unordered, ordered or ranked) and nested words, and various classes of graphs, such as cographs, graph classes of bounded tree-depth, those of bounded shrub-depth and n-partite cographs. Further, $\mathcal{L}$-EBSP remains preserved in the classes generated from the above by operations that are implementable using quantifier-free translation schemes. All of the aforementioned classes admit natural tree representations for their structures. We use this fact to show that the small and logically similar substructure of a large structure in any of these classes, can be computed in time linear in the size of the tree representation of the structure, giving linear time fixed parameter tractable (f.p.t.) algorithms for checking $\mathcal{L}$-definable properties of the large structure. We conclude by presenting a strengthening of $\mathcal{L}$-EBSP, that asserts "logical self-similarity at all scales" for a suitable notion of scale. We call this the *logical fractal* property and show that most of the classes mentioned above are indeed, logical fractals.

1998 ACM Subject Classification F.4.1 Model theory, F.4.3 Formal Languages, G.2.2 Graph theory

Keywords and phrases downward Löwenheim-Skolem theorem, trees, nested words, tree-depth, cographs, tree representation, translation schemes, composition lemma, f.p.t., logical fractal

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.37

1 Introduction

The downward Löwenheim-Skolem theorem is one of the earliest results of classical model theory. This theorem, first proved by Löwenheim in 1915 [22], states that if a first order (henceforth, FO) theory over a countable vocabulary has an infinite model, then it has a countable model. In the mid-1920s, Skolem came up with a more general statement: any structure $\mathfrak{A}$ over a countable vocabulary has a countable "FO-similar" substructure. Here, "FO-similarity" of two given structures means that the structures agree on all properties that can be expressed in FO. This result of Skolem was further generalized by Mal'tsev in 1936 [24], to what is considered as the modern statement of the downward Löwenheim-Skolem theorem: for any infinite cardinal κ, any structure $\mathfrak{A}$ over a countable vocabulary has an elementary substructure (an FO-similar substructure having additional properties) that has size at most κ. The downward Löwenheim-Skolem theorem is one of the most important

results of classical model theory, and indeed as Lindström showed in 1969 [21], FO is a maximal logic (having certain well-defined and reasonable closure properties) that satisfies this theorem, along with the (countable) compactness theorem.

The downward Löwenheim-Skolem theorem is intrinsically, a statement about infinite structures, and hence does not make sense in the finite when taken as is. While preservation and interpolation theorems from classical model theory have been actively studied over finite structures [16, 1, 26, 27, 3, 17], there is very little study of the downward Löwenheim-Skolem theorem (or adaptations of it) in the finite ([13, 34] seem to be the only studies of this theorem in the contexts of finite and pseudo-finite structures respectively). In this paper, we take a step towards addressing this issue. Specifically, we formulate a finitary analogue of the model-theoretic property contained in the downward Löwenheim-Skolem theorem, and show that classes of finite structures satisfying this analogue indeed abound in computer science. We call this analogue the *$\mathcal{L}$-equivalent bounded substructure property*, denoted $\mathcal{L}$-EBSP, where $\mathcal{L}$ is either FO or monadic second order logic (MSO). Intuitively, a class $\mathcal{S}$ of finite structures satisfies $\mathcal{L}$-EBSP if over $\mathcal{S}$, for each m, every structure $\mathfrak{A}$ contains a small substructure $\mathfrak{B}$ that is "$\mathcal{L}[m]$-similar" to $\mathfrak{A}$, where $\mathcal{L}[m]$ is the class of all sentences of $\mathcal{L}$ that have quantifier nesting depth at most m (Definition 3.1). In other words, $\mathfrak{B}$ and $\mathfrak{A}$ agree on all properties that can be described in $\mathcal{L}[m]$. The bound on the size of $\mathfrak{B}$ is given by a "witness function" that depends only on m (when $\mathcal{L}$ and $\mathcal{S}$ are fixed). It is easily seen that $\mathcal{L}$-EBSP has strong resemblance to the model-theoretic property contained in the downward Löwenheim-Skolem theorem, and can very well be seen as a finitary analogue of a version of this theorem that is "intermediate" between its versions by Skolem and Mal'tsev.

The motivation to define $\mathcal{L}$-EBSP came from our investigations over finite structures, of a generalization of the classical Łoś-Tarski preservation theorem from model theory, that was proved in [31]. This generalization, called the *generalized Łoś-Tarski theorem at level k*, denoted $\mathsf{GLT}(k)$, gives a semantic characterization, over arbitrary structures, of sentences in prenex normal form, whose quantifier prefixes are of the form $\exists^k\forall^*$, i.e. a sequence of k existential quantifiers followed by zero or more universal quantifiers. The Łoś-Tarski theorem is a special case of $\mathsf{GLT}(k)$ when k equals 0. Unfortunately, $\mathsf{GLT}(k)$ fails over all finite structures for all $k \geq 0$ (like most preservation theorems do [26]), and worse still, also fails for all $k \geq 2$, over the special classes of finite structures that are acyclic, of bounded degree, or of bounded tree-width, which were identified by Atserias, Dawar and Grohe [3] to satisfy the Łoś-Tarski theorem. This motivated the search for new (and possibly abstract) structural properties of classes of finite structures, that admit $\mathsf{GLT}(k)$ for each k. It is in this context that a version of $\mathcal{L}$-EBSP was first studied in [30]. The present paper takes that study much ahead. (Most of the results of this paper are from the author's Ph.D. thesis [28].)

Our results. We show that a variety of classes of finite structures of interest in computer science satisfy $\mathcal{L}$-EBSP, demonstrating that the latter property provides a unified framework, via logic, for studying these classes. The classes that we consider are broadly of two kinds: special kinds of labeled posets and special kinds of graphs. For the case of labeled posets, we show $\mathcal{L}$-EBSP holds for words, trees (of various kinds such as unordered, ordered, ranked, or "partially" ranked), and nested words over a finite alphabet, and all regular subclasses of these (Theorem 5.1). For each of these classes, we also show that $\mathcal{L}$-EBSP holds with computable witness functions. While words and trees have had a long history of studies in the literature, nested words are much recent [2], and have attracted a lot of attention as they admit a seamless generalization of the theory of regular languages, and are also closely connected with visibly pushdown languages. For the case of graphs, we show $\mathcal{L}$-EBSP holds

for a very general, and again very recently defined, class of graphs called *n-partite cographs*, and all hereditary subclasses of this class (Theorem 5.2). This class of graphs, introduced in [12], jointly generalizes the classes of cographs (which includes several interesting graph classes such as complete r-partite graphs, Turan graphs, cluster graphs, threshold graphs, etc.), graph classes of bounded tree-depth and those of bounded shrub-depth. Cographs have been well studied since the '80s and have been shown to admit fast algorithms for many decision and optimization problems that are hard in general [19]. Graph classes of bounded tree-depth and bounded shrub-depth are much more recently defined [25, 12] and have become particularly prominent in the context of investigating fixed parameter tractable (f.p.t.) algorithms for MSO model checking, that have *elementary dependence* on the size of the MSO sentence (which is the parameter) [11, 12]. This line of work seeks to identify classes of structures for which Courcelle-style *algorithmic meta-theorems* [15] hold, but with better dependence on the parameter than in the case of Courcelle's theorem (which is unavoidably non-elementary [10]). A different and important line of work shows that FO and MSO are equal in their expressive powers over graph classes of bounded tree-depth/shrub-depth [11, 8]. Since each of the graph classes mentioned above is a hereditary subclass of the class of n-partite cographs for some n, each of these satisfies $\mathcal{L}$-EBSP, further with computable witness functions, and further still, with even elementary witness functions in many cases.

We give methods to construct new classes of structures satisfying $\mathcal{L}$-EBSP from those known to satisfy the latter property. Specifically, we show that $\mathcal{L}$-EBSP remains preserved under a wide range of operations on structures, that have been well-studied in the literature: unary operations like complementation, transpose and the line graph operation, binary "sum-like" operations [23] such as disjoint union and join, and binary "product-like" operations that include the Cartesian, tensor, lexicographic and strong products. All of these are examples of operations that can be implemented using *quantifier-free translation schemes* [23]. We show that FO-EBSP is always closed under such operations, and MSO-EBSP is closed under such operations, provided that they are unary or sum-like (Theorem 5.3). In both cases, the computability/elementariness of witness functions is preserved under the operations.

As algorithmic consequences of the above results, we obtain linear time f.p.t. algorithms for model checking $\mathcal{L}$-definable properties of structures, over all of the aforementioned classes. Each of these classes, including those generated using the various operations, admits natural tree representations for its structures. Specifically, for any structure, a tree representation of it is such that any leaf node of the tree is labeled with a substructure (typically a simple one), while any internal node is labeled with an operation that produces a new structure upon being fed as input, the structures represented by the children of the internal node. Our f.p.t. algorithms utilize the fact that the input structures are given in the form of these tree representations, to perform model checking in time linear in the sizes of the representations. The techniques used in our algorithms are based on the *composition method* from model theory [7, 23, 33, 15]. This method, made prominent by the work of Feferman and Vaught [9], allows inferring the sentences true in a structure composed of simpler structures, from the latter structures. In our context, the method allows determining the "$\mathcal{L}[m]$-similarity class" of the output structure of an operation, from the multi-set of the $\mathcal{L}[m]$-similarity classes of the structures that are input to the operation. For operations having arbitrary finite arity, the $\mathcal{L}[m]$-similarity class of the output is determined only by a threshold number of appearances of each $\mathcal{L}[m]$-similarity class in the multi-set, with the threshold depending solely on m. These technical features, in conjunction with the fact that index of the $\mathcal{L}[m]$-similarity relation is always finite, facilitate us in algorithmically *generating* the "composition functions" uniformly for any operation for any given m. The functions for any operation, in turn, enable

doing the compositions in time linear in the arity of the operation. Using the latter fact, given a tree representation, we perform appropriate annotations, prunings and graftings in the tree iteratively, to produce in time linear in the size of the tree, a small subtree that represents a small $\mathcal{L}[m]$-similar substructure – the "kernel", in the f.p.t. parlance – of the structure represented by the given tree. Checking an $\mathcal{L}[m]$-definable property of the original structure then reduces to checking the same of the kernel. The above techniques have been incorporated into a single abstract result concerning tree representations (Theorem 4.2). Given that this result gives unified explanations for the good computational properties of many interesting classes, we believe it might be of independent interest.

Finally, we present a strengthened version of $\mathcal{L}$-EBSP, that turns out to be closely connected with the *fractal* property which has been extensively studied in mathematics, and in connection with a variety of natural phenomena [4]. Fractals are classes of mathematical structures that exhibit self-similarity at all scales. That is, every structure in the class contains a similar (in some technical sense) substructure at every scale of sizes less than the size of the structure. In this light, we observe that $\mathcal{L}$-EBSP indeed asserts "logical self-similarity" at "small scales". We formulate a strengthening of $\mathcal{L}$-EBSP, that asserts logical self-similarity at all scales, for a suitable notion of scale (Definition 6.1). We call this the *logical fractal* property, and call a class satisfying this property a *logical fractal*. It turns out that all of the aforementioned examples of poset and graph classes are logical fractals (Proposition 6.2). Further, all of the aforementioned operations on structures preserve the logical fractal property. The latter property thus appears to be naturally arising in diverse interesting settings of computer science. This suggests that adapting concepts from (mathematical) fractal theory (for instance, fractal dimension) to their logical counterparts can yield useful notions for finite model theory.

Paper organization. In Section 2, we introduce terminology and notation, and recall relevant notions from the literature used in the paper. In Section 3, we define $\mathcal{L}$-EBSP formally and show that it holds for the class of "partially" ranked trees, which are trees in which some subset of nodes are constrained to have degrees given by a ranking function. We use this special class as a setting to illustrate our techniques, that we lift to tree representations of structures in Section 4. In Section 5, we give applications of our abstract result to show $\mathcal{L}$-EBSP and the existence of linear time f.p.t. algorithms for $\mathcal{L}$-model checking, in various concrete settings, specifically those of posets and graphs mentioned earlier, and also classes constructed using various well-studied operations. We present the notion of logical fractals in Section 6, and conclude with open questions in Section 7.

2 Terminology and preliminaries

We assume familiarity with standard notions and notation of first order logic (FO) and monadic second order logic (MSO) [20]. By $\mathcal{L}$, we mean either FO or MSO. We consider only finite vocabularies, represented by τ or ν, that contain only *predicate symbols* (and no constant or function symbols), unless explicitly stated otherwise. All predicate symbols are assumed to have *positive* arity. We denote by $\mathcal{L}(\tau)$ the set of all $\mathcal{L}$ formulae over τ (and refer to these simply as $\mathcal{L}$ formulae, when τ is clear from context). A sequence $(x_1, \ldots, x_k)$ of variables is written as $\bar{x}$. A formula φ whose free variables are among $\bar{x}$, is denoted as $\varphi(\bar{x})$. Free variables are always *first order*. A formula with no free variables is called a *sentence*. The *rank* of an $\mathcal{L}$ formula is the maximum number of quantifiers (first order as well as second order) that appear along any path from the root to a leaf in the parse tree of the formula.

Finally, *a notion or result stated for $\mathcal{L}$ means that the notion or result respectively, is stated for both FO and MSO.*

Standard notions of τ-structures (denoted $\mathfrak{A}, \mathfrak{B}$ etc.; we refer to these simply as structures when τ is clear from context), substructures (denoted $\mathfrak{A} \subseteq \mathfrak{B}$) and extensions are used throughout the paper (see [20]). We assume all structures to be *finite*. As in [20], by substructures, we always mean *induced* substructures. Given a structure $\mathfrak{A}$, we use $\mathsf{U}_{\mathfrak{A}}$ to denote the universe of $\mathfrak{A}$, and $|\mathfrak{A}|$ to denote its cardinality. We denote by $\mathfrak{A} \cong \mathfrak{B}$ that $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$, and by $\mathfrak{A} \hookrightarrow \mathfrak{B}$ that $\mathfrak{A}$ is isomorphically embeddable in $\mathfrak{B}$. For an $\mathcal{L}$ sentence φ, we denote by $\mathfrak{A} \models \varphi$ that $\mathfrak{A}$ is a model of φ. We denote classes of structures by $\mathcal{S}$, possibly with subscripts, and assume these to be *closed under isomorphisms*.

Let $\mathbb{N}$ and $\mathbb{N}_+$ denote the natural numbers including zero and excluding zero respectively. Given $m \in \mathbb{N}$ and a τ-structure $\mathfrak{A}$, denote by $\mathsf{Th}_{m,\mathcal{L}}(\mathfrak{A})$ the set of all $\mathcal{L}(\tau)$ sentences of rank at most m, that are true in $\mathfrak{A}$. Given a τ-structure $\mathfrak{B}$, we say that $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathcal{L}[m]$-*equivalent*, denoted $\mathfrak{A} \equiv_{m,\mathcal{L}} \mathfrak{B}$ if $\mathsf{Th}_{m,\mathcal{L}}(\mathfrak{A}) = \mathsf{Th}_{m,\mathcal{L}}(\mathfrak{B})$. Given a class $\mathcal{S}$ of structures and $m \in \mathbb{N}$, we let $\Delta_{\mathcal{S},\mathcal{L},m}$ denote the set of all equivalence classes of the $\equiv_{m,\mathcal{L}}$ relation over $\mathcal{S}$. We denote by $\Lambda_{\mathcal{S},\mathcal{L}} : \mathbb{N} \to \mathbb{N}$ a fixed computable function with the property that $\Lambda_{\mathcal{S},\mathcal{L}}(m) \geq |\Delta_{\mathcal{S},\mathcal{L},m}|$. It is known that $\Lambda_{\mathcal{S},\mathcal{L}}$ always exists (see Proposition 7.5 in [20]). The notion of $\equiv_{m,\mathcal{L}}$ has a characterization using *Ehrenfeucht-Fräissé* (EF) games for $\mathcal{L}$. We point the reader to Chapters 3 and 7 of [20] for results concerning these games.

We recall the notion of translation schemes from the literature [23] (known in the literature by different names, like *interpretations, transductions*, etc). Let τ and ν be given vocabularies, and $t \geq 1$ be a natural number. Let $\bar{x}_0$ be a fixed t-tuple of first order variables, and for each relation $R \in \nu$ of arity $\#R$, let $\bar{x}_R$ be a fixed $(t \times \#R)$-tuple of first order variables. A $(t, \tau, \nu, \mathcal{L})$-*translation scheme* $\Xi = (\xi, (\xi_R)_{R \in \nu})$ is a sequence of formulas of $\mathcal{L}(\tau)$ such that the free variables of ξ are among those in $\bar{x}_0$, and for $R \in \nu$, the free variables of ξ_R are among those in $\bar{x}_R$. When t, ν and τ are clear from context, we call Ξ simply a translation scheme, and t the *dimension* of Ξ. The translation scheme Ξ defines a map from τ-structures to ν-structures [23]. Abusing notation slightly, we denote this map as Ξ again, and for a class $\mathcal{S}$ of τ-structures, let $\Xi(\mathcal{S})$ denote the class $\{\Xi(\mathfrak{A}) \mid \mathfrak{A} \in \mathcal{S}\}$.

Given a class $\mathcal{S}$ of structures, we say that the model checking problem for $\mathcal{L}$ over $\mathcal{S}$, denoted $\mathsf{MC}(\mathcal{L}, \mathcal{S})$, is *fixed parameter tractable* [15], in short *f.p.t.*, if there exists an algorithm Alg that when given as input an $\mathcal{L}$ sentence φ of rank m, and a structure $\mathfrak{A} \in \mathcal{S}$, decides if $\mathfrak{A} \models \varphi$, in time $f(m) \cdot |\mathfrak{A}|^c$, where $f : \mathbb{N} \to \mathbb{N}$ is some computable function and c is a constant. In this case, we say Alg is an *f.p.t. algorithm* for $\mathsf{MC}(\mathcal{L}, \mathcal{S})$. We say Alg is a *linear time* f.p.t. algorithm for $\mathsf{MC}(\mathcal{L}, \mathcal{S})$ if it is f.p.t. for $\mathsf{MC}(\mathcal{L}, \mathcal{S})$ and runs in time $f(m) \cdot |\mathfrak{A}|$ where as before, f is a computable function and m is the rank of the input sentence.

The k-*fold* exponential function $\mathsf{exp}(n, k)$ is the function given inductively as: $\mathsf{exp}(n, 0) = n$ and $\mathsf{exp}(n, l) = 2^{\mathsf{exp}(n, l-1)}$ for $0 \leq l \leq k$. We call a function $f : \mathbb{N} \to \mathbb{N}$ *elementary* if there exists k such that $f(n) = O(\mathsf{exp}(n, k))$, and call it *non-elementary* if it is not elementary.

Finally, we use the following standard abbreviations throughout the paper: 'w.l.o.g' for 'without loss of generality', 'iff' for 'if and only if', and 'resp.' for 'respectively'.

3 The $\mathcal{L}$-Equivalent Bounded Substructure Property

▶ **Definition 3.1** ($\mathcal{L}$-EBSP($\mathcal{S}$))**.** Let $\mathcal{S}$ be a class of structures and $\mathcal{L}$ be either FO or MSO. We say that $\mathcal{S}$ satisfies the $\mathcal{L}$-*equivalent bounded substructure property*, abbreviated $\mathcal{L}$-EBSP($\mathcal{S}$) *is true* (alternatively, $\mathcal{L}$-EBSP($\mathcal{S}$) *holds*), if there exists an increasing function $\theta_{(\mathcal{S},\mathcal{L})} : \mathbb{N} \to \mathbb{N}$ such that for each $m \in \mathbb{N}$ and each structure $\mathfrak{A}$ of $\mathcal{S}$, there exists a structure $\mathfrak{B}$ such that

(i) $\mathfrak{B} \in \mathcal{S}$, (ii) $\mathfrak{B} \subseteq \mathfrak{A}$, (iii) $|\mathfrak{B}| \leq \theta_{(\mathcal{S},\mathcal{L})}(m)$, and (iv) $\mathfrak{B} \equiv_{m,\mathcal{L}} \mathfrak{A}$. The conjunction of these four conditions is denoted as $\mathcal{L}$-EBSP-condition$(\mathcal{S}, \mathfrak{A}, \mathfrak{B}, m, \theta_{(\mathcal{S},\mathcal{L})})$. We call $\theta_{(\mathcal{S},\mathcal{L})}$ a *witness function for* $\mathcal{L}$-EBSP$(\mathcal{S})$, and say $\mathcal{L}$-EBSP$(\mathcal{S})$ *holds with a witness function* $\theta_{(\mathcal{S},\mathcal{L})}$.

We present below two simple examples of classes satisfying $\mathcal{L}$-EBSP.

1. Let $\mathcal{S}$ be the class of all τ-structures, where all predicates in τ are unary. By a simple FO-EF game argument, we see that FO-EBSP$(\mathcal{S})$ holds with a witness function $\theta_{(\mathcal{S},\mathrm{FO})} : \mathbb{N} \to \mathbb{N}$ given by $\theta_{(\mathcal{S},\mathrm{FO})}(m) = m \cdot 2^{|\tau|}$. In more detail: given $\mathfrak{A} \in \mathcal{S}$, associate exactly one of $2^{|\tau|}$ colors with each element a of $\mathfrak{A}$, where the colour gives the valuation of all predicates of τ for a in $\mathfrak{A}$. Then consider $\mathfrak{B} \subseteq \mathfrak{A}$ such that for each colour c, if $A_c = \{a \mid a \in \mathsf{U}_{\mathfrak{A}},\ a \text{ has colour } c \text{ in } \mathfrak{A}\}$, then $A_c \subseteq \mathsf{U}_{\mathfrak{B}}$ if $|A_c| < m$, else $|A_c \cap \mathsf{U}_{\mathfrak{B}}| = m$. It is easy to verify that FO-EBSP-condition$(\mathcal{S}, \mathfrak{A}, \mathfrak{B}, m, \theta_{(\mathcal{S},\mathrm{FO})})$ holds. By a similar MSO-EF game argument, MSO-EBSP$(\mathcal{S})$ holds with a witness function given by $\theta_{(\mathcal{S},\mathrm{MSO})}(m) = m \cdot 2^{(|\tau|+m)}$.
2. Let $\mathcal{S}$ be the class of disjoint unions of undirected paths. It can be easily shown that for any m, any two paths of length $\geq p = 3^m$ are FO$[m]$-equivalent. Let $\mathfrak{A} = \bigsqcup_{n \geq 0} i_n \cdot P_n$ where P_n denotes the path of length n, $i_n \cdot P_n$ denotes the disjoint union of i_n copies of P_n, and $\bigsqcup$ denotes disjoint union. For $n < p$, let j_n be such that $j_n = i_n$ if $i_n < m$ and $j_n = m$ if $i_n \geq m$. For $n = p$, let $j_n = h = \sum_{r \geq p} i_r$ if $h < m$, else $j_n = m$. If $\mathcal{B} = \bigsqcup_{n=1}^{n=p} j_n \cdot P_n$, then by an FO-EF game argument, $\mathfrak{B}$ satisfies FO-EBSP-condition$(\mathcal{S}, \mathfrak{A}, \mathfrak{B}, m, \theta_{(\mathcal{S},\mathrm{FO})})$ where $\theta_{(\mathcal{S},\mathrm{FO})}(m) = \sum_{n=0}^{n=p} m \cdot n$. Then FO-EBSP$(\mathcal{S})$ holds with a witness function $\theta_{(\mathcal{S},\mathrm{FO})}$.

3.1 Partially ranked trees satisfy $\mathcal{L}$-EBSP

In this subsection, we show that the class of ordered "partially" ranked trees satisfies $\mathcal{L}$-EBSP with computable witness functions, as well as admits a linear time f.p.t. algorithm for model checking $\mathcal{L}$ sentences. This setting illustrates our reasoning and techniques that we lift in Section 4 to the more abstract setting of tree representations of structures.

An *unlabeled unordered tree* is a finite poset $P = (A, \leq)$ with a unique minimal element (called "root"), such that for each $c \in A$, the set $\{b \mid b \leq c\}$ is totally ordered by $\leq$. Informally speaking, the Hasse diagram of P is an inverted (graph-theoretic) tree. We call A as the set of *nodes* of P. We use the standard notions of leaf, internal node, ancestor, descendent, parent, child, degree, height, and subtree in connection with trees. (We clarify that by height, we mean the maximum distance between the root and any leaf of the tree, as against the "number of levels" in the tree.) An *unlabeled ordered* tree is a pair $O = (P, \lesssim)$ where P is an unlabeled unordered tree and $\lesssim$ is a binary relation that imposes a linear order on the children of any internal node of P. Unless explicitly stated otherwise, we always consider our trees to be *ordered*. It is clear that the above mentioned notions in connection with unordered trees can be adapted for ordered trees. Given a countable alphabet Σ, a *tree over* Σ, also called a Σ*-tree*, or simply *tree* when Σ is clear from context, is a pair (O, λ) where O is an unlabeled tree and $\lambda : A \to \Sigma$ is a labeling function, where A is the set of nodes of O. We denote Σ-trees by $\mathsf{s}, \mathsf{t}, \mathsf{x}, \mathsf{y}, \mathsf{u}, \mathsf{v}$ or z, possibly with numbers as subscripts. Given a tree t, we denote the root of t as $\mathsf{root}(\mathsf{t})$. For a node a of t, we denote the subtree of t rooted at a as $\mathsf{t}_{\geq a}$, and the subtree of t obtained by deleting $\mathsf{t}_{\geq a}$ from t, as $\mathsf{t} - \mathsf{t}_{\geq a}$. Given a tree s and a non-root node a of t, the *replacement of* $\mathsf{t}_{\geq a}$ *with* s *in* t, denoted $\mathsf{t}\,[\mathsf{t}_{\geq a} \mapsto \mathsf{s}]$, is a tree defined as follows. Assume w.l.o.g. that s and t have disjoint sets of nodes. Let c be the parent of a in t. Then $\mathsf{t}\,[\mathsf{t}_{\geq a} \mapsto \mathsf{s}]$ is defined as the tree obtained by deleting $\mathsf{t}_{\geq a}$ from t to get a tree t', and inserting (the root of) s at the same position among the children of c in t', as the position of a among the children of c in t. For s and t as just mentioned, suppose the roots of both these trees have the same label. Then the *merge of* s *with* t, denoted $\mathsf{t} \odot \mathsf{s}$, is

defined as the tree obtained by deleting root(s) from s and concatenating the sequence of subtrees hanging at root(s) in s, to the sequence of subtrees hanging at root(t) in t. Thus the children of root(s) in s are the "new" children of root(t), and appear "after" the "old" children of root(t), and in the order they appear in s.

Fix a finite alphabet Σ, and let $\Sigma_{\text{rank}} \subseteq \Sigma$. Let $\rho : \Sigma_{\text{rank}} \to \mathbb{N}_+$ be a fixed function. We say a Σ-tree $\mathsf{t} = (O, \lambda)$ is *partially ranked by* (Σ_{rank}, ρ) if for any node a of t, if $\lambda(a) \in \Sigma_{\text{rank}}$, then the number of children of a in t is exactly $\rho(\lambda(a))$. Observe that the case of $\Sigma_{\text{rank}} = \Sigma$ corresponds to the notion of ranked trees that are well-studied in the literature [5]. Let Partially-ranked-trees$(\Sigma, \Sigma_{\text{rank}}, \rho)$ be the class of all ordered Σ-trees partially ranked by $(\Sigma_{\text{rank}}, \rho)$. The central result of this section is now as stated below.

▶ **Proposition 3.2.** *Given Σ, a subset Σ_{rank} of Σ and $\rho : \Sigma_{rank} \to \mathbb{N}_+$, let $\mathcal{S}$ be the class* Partially-ranked-trees$(\Sigma, \Sigma_{rank}, \rho)$. *Then the following are true:*

1. *$\mathcal{L}$-EBSP$(\mathcal{S})$ holds with a computable witness function. Further, any witness function is necessarily non-elementary.*
2. *There is a linear time f.p.t. algorithm for* MC$(\mathcal{L}, \mathcal{S})$.

We prove the two parts of the above result separately. In the remainder of this section, we fix $\mathcal{L}$, and also fix $\mathcal{S}$ to be the class Partially-ranked-trees$(\Sigma, \Sigma_{\text{rank}}, \rho)$. Given these fixings, we denote $\Delta_{\mathcal{S},\mathcal{L},m}$ (the set of equivalence classes of the $\equiv_{m,\mathcal{L}}$ relation over $\mathcal{S}$) simply as Δ_m, and denote $\Lambda_{\mathcal{S},\mathcal{L}}(m)$ (see Section 2 for the definition of $\Lambda_{\mathcal{S},\mathcal{L}}(m)$) simply as $\Lambda(m)$.

▶ Remark. As mentioned in the introduction, our techniques to prove Proposition 3.2, are based on the "composition method" from model theory. One can also adopt automata based techniques to prove Proposition 3.2. Specifically, part (1) can be shown using the "downward direction" of the pumping lemma for such trees [5], while part (2) can be shown using the automata based approach described in [32]. Though, admittedly, the automata based approach is easier than the composition method for this special case of partially ranked trees, the machinery that we develop in this (illustrative) section admits a direct and seamless lifting to the abstract setting of tree representations that we consider in the next section. The abstract setting enables us to uniformly show our $\mathcal{L}$-EBSP and f.p.t. results for a wide range of interesting concrete instantiations that we present in Section 5.

Towards the proof of Proposition 3.2, we first present an $\mathcal{L}$-composition lemma for partially ranked trees, that is the "functional form" of a Feferman-Vaught (FV) style $\mathcal{L}$-composition lemma for these trees (See Appendix A for an FV-style $\mathcal{L}$-composition lemma for the more general case of ordered trees.) Recall that $\mathcal{S} =$ Partially-ranked-trees$(\Sigma, \Sigma_{\text{rank}}, \rho)$.

▶ **Lemma 3.3** (Composition lemma for partially ranked trees)**.** *For each $\sigma \in \Sigma$ and $m \geq 3$, there exists a function $f_{\sigma,m} : (\Delta_m)^{\rho(\sigma)} \to \Delta_m$ if $\sigma \in \Sigma_{rank}$, and functions $f_{\sigma,m,i} : (\Delta_m)^i \to \Delta_m$ for $i \in \{1, 2\}$ if $\sigma \in \Sigma \setminus \Sigma_{rank}$, with the following properties: Let $\mathsf{t} = (O, \lambda) \in \mathcal{S}$ and a be an internal node of t such that $\lambda(a) = \sigma$, and the children of a in t are $b_1, \ldots, b_n$. Let δ_i be the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{t}_{\geq b_i}$ for $i \in \{1, \ldots, n\}$, and let δ be the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{t}_{\geq a}$.*

1. *If $\sigma \in \Sigma_{rank}$ (whereby $n = \rho(\sigma)$), then $\delta = f_{\sigma,m}(\delta_1, \ldots, \delta_n)$.*
2. *If $\sigma \in \Sigma \setminus \Sigma_{rank}$, then δ is given as follows: For $k \in \{1, \ldots, n-1\}$, let $\chi_{k+1} = f_{\sigma,m,2}(\chi_k, \delta_{k+1})$ where $\chi_1 = f_{\sigma,m,1}(\delta_1)$. Then $\delta = \chi_n$.*

A useful corollary of this lemma is as below.

▶ **Corollary 3.4.** *The following are true for $m \geq 3$.*

1. *Given trees s, t and a non-root node a of t, let $\mathsf{z} = \mathsf{t}[\mathsf{t}_{\geq a} \mapsto \mathsf{s}]$. If $\mathsf{s} \equiv_{m,\mathcal{L}} \mathsf{t}_{\geq a}$, then $\mathsf{z} \equiv_{m,\mathcal{L}} \mathsf{t}$.*

2. *Let* $\mathsf{s}_1, \mathsf{s}_2, \mathsf{t}$ *be given trees such that the labels of their roots are the same, and belong to* $\Sigma \setminus \Sigma_{rank}$*. Suppose* $\mathsf{z}_i = \mathsf{s}_i \odot \mathsf{t}$ *for* $i \in \{1, 2\}$*. If* $\mathsf{s}_1 \equiv_{m,\mathcal{L}} \mathsf{s}_2$*, then* $\mathsf{z}_1 \equiv_{m,\mathcal{L}} \mathsf{z}_2$*.*
3. *Let* $\mathsf{s}_1, \mathsf{s}_2$ *be given trees such that the labels of their roots are the same, and belong to* $\Sigma \setminus \Sigma_{rank}$*. For* $i \in \{1, 2\}$*, given* t_i*, let* z_i *be the tree obtained from* s_i *by adding* t_i *as the (new) "last" child subtree of the root of* s_i*. If* $\mathsf{s}_1 \equiv_{m,\mathcal{L}} \mathsf{s}_2$ *and* $\mathsf{t}_1 \equiv_{m,\mathcal{L}} \mathsf{t}_2$*, then* $\mathsf{z}_1 \equiv_{m,\mathcal{L}} \mathsf{z}_2$*.*

Proof sketch for part (1) of Proposition 3.2: The first half follows from the following "reduction" lemma for trees. The second half follows from the fact that even over Σ-words (Σ-labeled linear orders), the index of the $\equiv_{m,\mathcal{L}}$ relation depends non-elementarily on m [10].

▶ **Lemma 3.5.** *There exist computable functions* $\eta_1, \eta_2 : \mathbb{N} \to \mathbb{N}$ *such that for each* $\mathsf{t} \in \mathcal{S}$ *and* $m \in \mathbb{N}$*, the following hold:*

1. (Degree reduction) *There exists a subtree* s_1 *of* t *in* $\mathcal{S}$*, of degree* $\leq \eta_1(m)$*, such that* (i) *the roots of* s_1 *and* t *are the same, and* (ii) $\mathsf{s}_1 \equiv_{m,\mathcal{L}} \mathsf{t}$*.*
2. (Height reduction) *There exists a subtree* s_2 *of* t *in* $\mathcal{S}$*, of height* $\leq \eta_2(m)$*, such that* (i) *the roots of* s_2 *and* t *are the same, and* (ii) $\mathsf{s}_2 \equiv_{m,\mathcal{L}} \mathsf{t}$*.*

Proof. For a finite subset X of $\mathbb{N}$, let $\max(X)$ denote the maximum element of X.

(1) For $n \geq 3$, define $\eta_1(n) = \max(\{\rho(\sigma) \mid \sigma \in \Sigma_{\text{rank}}\} \cup \{3\}) \times \Lambda(n)$. For $n < 3$, define $\eta_1(n) = \eta_1(3)$. We prove this part for $m \geq 3$; then it follows that this part is also true for $m < 3$ (by taking s_1 for the $m = 3$ case as s_1 for the $m < 3$ case).
Given $m \geq 3$, let $p = \eta_1(m)$. If t has degree $\leq p$, then putting $\mathsf{s}_1 = \mathsf{t}$, we are done. Else, some node a of t has degree $n > p$. Clearly then $\lambda(a) \notin \Sigma_{\text{rank}}$. Let $\mathsf{z} = \mathsf{t}_{\geq a}$ and let $a_1, \ldots, a_n$ be the (ascending) sequence of children of $\mathsf{root}(\mathsf{z})$ in z. For $1 \leq j \leq n$, let $\mathsf{x}_{1,j}$, resp. $\mathsf{y}_{j+1,n}$, be the subtree of z obtained from z by deleting the subtrees rooted at $a_{j+1}, \ldots, a_n$, resp. deleting the subtrees rooted at $a_1, a_2, \ldots, a_j$. Then $\mathsf{z} = \mathsf{x}_{1,n} = \mathsf{x}_{1,j} \odot \mathsf{y}_{j+1,n}$ for $1 \leq j < n$. Let $g : \{1, \ldots, n\} \to \Delta_m$ be such that $g(j)$ is the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{x}_{1,j}$. Since $n > p$, there exist $j, k \in \{1, \ldots, n\}$ such that $j < k$ and $g(j) = g(k)$, i.e. $\mathsf{x}_{1,j} \equiv_{m,\mathcal{L}} \mathsf{x}_{1,k}$. If $k < n$, then let $\mathsf{z}_1 = \mathsf{x}_{1,j} \odot \mathsf{y}_{k+1,n}$, else let $\mathsf{z}_1 = \mathsf{x}_{1,j}$. Then by Corollary 3.4, $\mathsf{z}_1 \equiv_{m,\mathcal{L}} \mathsf{z}$. Let t_1 be the subtree of t in $\mathcal{S}$ given by $\mathsf{t}_1 = \mathsf{t}[\mathsf{z} \mapsto \mathsf{z}_1]$. By Corollary 3.4 again, $\mathsf{t}_1 \equiv_{m,\mathcal{L}} \mathsf{t}$. Observe that t_1 has strictly lesser size than t. Recursing on t_1, we are eventually done.

(2) For $n \geq 3$, define $\eta_2(n) = \Lambda(n) + 1$. For $n < 3$, define $\eta_2(n) = \eta_2(3)$. As before, it suffices to prove this part for $m \geq 3$.
Given $m \geq 3$, let $p = \eta_2(m)$. If t has height $\leq p$, then putting $\mathsf{s}_2 = \mathsf{t}$, we are done. Else, there is a path from the root of t to some leaf of t, whose length is $> p$. Let A be the set of nodes appearing along this path. Let $h : A \to \Delta_m$ be such that for each $a \in A$, $h(a)$ is the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{t}_{\geq a}$. Since $|A| > p$, there exist distinct nodes $a, b \in A$ such that a is an ancestor of b in t, $a \neq \mathsf{root}(\mathsf{t})$, and $h(a) = h(b)$. Let $\mathsf{t}_2 = \mathsf{t}[\mathsf{t}_{\geq a} \mapsto \mathsf{t}_{\geq b}]$; then t_2 is a subtree of t in $\mathcal{S}$. Since $h(a) = h(b)$, $\mathsf{t}_{\geq a} \equiv_{m,\mathcal{L}} \mathsf{t}_{\geq b}$. By Corollary 3.4, we get $\mathsf{t}_2 \equiv_{m,\mathcal{L}} \mathsf{t}$. Note that t_2 has strictly lesser size than t. Recursing on t_2, we are eventually done.

◀

Proof sketch for part (2) of Proposition 3.2: The following result contains the core argument for the proof of this part. The first part of Lemma 3.6 gives an algorithm to generate the "composition" functions of Lemma 3.3, uniformly for $m \geq 3$. This algorithm is in turn used in the second part of Lemma 3.6 to get a "linear time" version of Lemma 3.5.

▶ **Lemma 3.6.** *There exist computable functions* $\eta_3, \eta_4, \eta_5 : \mathbb{N} \to \mathbb{N}$ *and algorithms* $\mathsf{Generate\text{-}functions}(m)$*,* $\mathsf{Reduce\text{-}degree}(\mathsf{t}, m)$ *and* $\mathsf{Reduce\text{-}height}(\mathsf{t}, m)$ *such that for* $m \geq 3$*,*

1. Generate-functions(m) *generates in time* $\eta_3(m)$*, the functions* $f_{\sigma,m}$ *if* $\sigma \in \Sigma_{rank}$ *and* $f_{\sigma,m,i}$ *for* $i \in \{1,2\}$ *if* $\sigma \in \Sigma \setminus \Sigma_{rank}$*, that satisfy the properties mentioned in Lemma 3.3.*
2. *For* $\mathsf{t} \in \mathcal{S}$*,* Reduce-degree$(\mathsf{t}, m)$ *computes the subtree* s_1 *of* t *as given by Lemma 3.5, in time* $\eta_4(m) \cdot |\mathsf{t}|$*. Likewise,* Reduce-height(t, m) *computes the subtree* s_2 *of* t *as given by Lemma 3.5, in time* $\eta_5(m) \cdot |\mathsf{t}|$*.*

Proof Sketch.

(1) We first observe that the $\mathcal{L}$-SAT problem is decidable over $\mathcal{S}$ – since $\mathcal{L}$-EBSP$(\mathcal{S})$ holds with a computable witness function (by Proposition 3.2(1)), if an $\mathcal{L}$ sentence has a model in $\mathcal{S}$, it also has a model of size bounded by a computable function of its rank.

Generate-functions(m)

1. Create a list $\mathcal{L}[\mathsf{m}]$-classes of the $\equiv_{m,\mathcal{L}}$ classes over $\mathcal{S}$. This is done as follows:
   a. Given the inductive definition of $\mathcal{L}[m]$, there is an algorithm $\mathcal{P}(m)$ that enumerates $\mathcal{L}[m]$ sentences $\varphi_1, \varphi_2, \ldots, \varphi_n$ such that each φ_i captures some class in $\Delta_{\mathsf{All},\mathcal{L},m}$ (the set of equivalence classes of the $\equiv_{m,\mathcal{L}}$ relation over all finite structures), and conversely, each class of $\Delta_{\mathsf{All},\mathcal{L},m}$ is captured by some φ_i. First invoke $\mathcal{P}(m)$ to get the φ_is.
   b. For each $i \in \{1, \ldots, n\}$, if φ_i is satisfiable over $\mathcal{S}$ (whereby it represents some equivalence class of the $\equiv_{m,\mathcal{L}}$ relation over $\mathcal{S}$), then put it in $\mathcal{L}[\mathsf{m}]$-classes, else discard it. (We interchangeably regard $\mathcal{L}[\mathsf{m}]$-classes as a list of $\mathcal{L}[m]$ sentences or a list of $\equiv_{m,\mathcal{L}}$ classes.)
2. For $\sigma \in \Sigma_{\text{rank}}$ and $d = \rho(\sigma)$, generate $g_{\sigma,m} : (\mathcal{L}[\mathsf{m}]\text{-classes})^d \to \mathcal{L}[\mathsf{m}]$-classes as follows. Given $\xi_i \in \mathcal{L}[\mathsf{m}]$-classes for $i \in \{1, \ldots, d\}$, find models s_i for ξ_i in $\mathcal{S}$. Let s be the tree obtained by making $\mathsf{s}_1, \ldots, \mathsf{s}_n$ as the child subtrees (and in that sequence) of a new root node labeled with σ. Find out $\xi \in \mathcal{L}[\mathsf{m}]$-classes of which s is a model. Then define $g_{\sigma,m}(\xi_1, \ldots, \xi_d) = \xi$. Generate $g_{\sigma,m,1} : \mathcal{L}[\mathsf{m}]\text{-classes} \to \mathcal{L}[\mathsf{m}]$-classes similarly.
3. For $\sigma \in \Sigma \setminus \Sigma_{\text{rank}}$, generate $g_{\sigma,m,2} : (\mathcal{L}[\mathsf{m}]\text{-classes})^2 \to \mathcal{L}[\mathsf{m}]$-classes as follows. For $\xi_1, \xi_2 \in \mathcal{L}[\mathsf{m}]$-classes, find models s_1 and s_2 resp. in $\mathcal{S}$. such that the root of s_1 is labeled with σ (this condition on the root can be captured by an FO sentence). If no s_1 is found, then define $g_{\sigma,m,2}(\xi_1, \xi_2) = \xi_{\text{default}}$ where the latter is some fixed element of $\mathcal{L}[\mathsf{m}]$-classes. Else, let v_{ξ_1,ξ_2} be the tree obtained adding s_2 as the (new) "last" child subtree of the root of s_1. Find out $\xi \in \mathcal{L}[\mathsf{m}]$-classes of which v_{ξ_1,ξ_2} is a model. Define $g_{\sigma,m,2}(\xi_1, \xi_2) = \xi$.

It is clear that there exists a computable function $\eta_3 : \mathbb{N} \to \mathbb{N}$ such that the running time of Generate-functions(m) is at most $\eta_3(m)$. We now claim that $g_{\sigma,m}$ and $g_{\sigma,m,i}$ generated by Generate-functions(m) indeed satisfy the composition properties of Lemma 3.3, whereby they can be indeed taken as $f_{\sigma,m}$ and $f_{\sigma,m,i}$ appearing in the latter lemma. That $g_{\sigma,m}$ and $g_{\sigma,m,1}$ satisfy the composition properties is easy to see using Corollary 3.4. To reason for $g_{\sigma,m,2}$, consider a tree t whose root is labeled with σ, and which has say 3 children $a_1, \ldots, a_3$ (and in that sequence) such that the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{t}_{\geq a_i}$ is δ_i for $1 \leq i \leq 3$. Consider the subtrees x and y of t defined as $\mathsf{x} = \mathsf{t} - \mathsf{t}_{\geq a_3}$ and $\mathsf{y} = \mathsf{x} - \mathsf{x}_{\geq a_2}$. Let δ_4 and δ_5 be resp. the $\equiv_{m,\mathcal{L}}$ classes of x and y. Now consider the trees $\mathsf{v}_{\delta_5,\delta_2}$ and $\mathsf{v}_{\delta_4,\delta_3}$ which are guaranteed to be found (since indeed x and y *are* trees each of whose roots is labeled with σ). By Corollary 3.4, $\mathsf{x} \equiv_{m,\mathcal{L}} \mathsf{v}_{\delta_5,\delta_2}$ and $\mathsf{t} \equiv_{m,\mathcal{L}} \mathsf{v}_{\delta_4,\delta_3}$. Whereby, the $\equiv_{m,\mathcal{L}}$ class of x is $\delta_4 = g_{\sigma,m,2}(\delta_5, \delta_2)$ and that of t is $\delta = g_{\sigma,m,2}(\delta_4, \delta_3)$. Observe that δ_5 is indeed $g_{\sigma,m,1}(\delta_1)$.

(2) Reduce-degree(t, m)

1. Call Generate-functions(m) that returns the "composition" functions $f_{\sigma,m}$ and $f_{\sigma,m,i}$, and also gives the list $\mathcal{L}$[m]-classes as described above.
2. Using the composition functions, construct bottom-up in t, the function Colour : Nodes(t) $\to \mathcal{L}$[m]-classes such that for each node a of t, Colour(a) is the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{t}_{\geq a}$.
3. For η_1 as given by Lemma 3.5, if the degree of t is $\leq \eta_1(m)$, then return t.
4. Else, let a be a node of t of degree $n > \eta_1(m)$. Let $\mathsf{x} = \mathsf{t}_{\geq a}$.
5. For each $\delta \in \mathcal{L}$[m]-classes, do the following:
   a. Let $a_1, \ldots, a_n$ be the children of a in x. For $k \in \{1, \ldots, n\}$, let $\mathsf{x}_{1,k}$ be the subtree of x obtained by deleting the subtrees rooted at $a_{k+1}, \ldots, a_n$. Let $g : \{1, \ldots, n\} \to \mathcal{L}$[m]-classes be such that $g(i)$ is the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{x}_{1,k}$.
   b. If δ appears in the range of g, then let i, j be resp. the least and greatest indices in $\{1, \ldots, n\}$ such that $g(i) = g(j) = \delta$. Let y be the subtree of x obtained by deleting the subtrees rooted at $a_{i+1}, \ldots, a_j$. Set $\mathsf{x} := \mathsf{y}$.
6. Set $\mathsf{t} := \mathsf{t}[\mathsf{t}_{\geq a} \mapsto \mathsf{x}]$ and go to step 3.

Reasoning similarly as for Lemma 3.5(1), we can verify that Reduce-degree(t, m) indeed returns the desired subtree s_1 of t. The time taken to compute Colour is linear in $|\mathsf{t}|$, while that for computing g is linear in the degree of a, whereby the time taken to reduce the degree of a node a in any iteration of the loop, is $O(\Lambda(m) \cdot \text{degree}(a))$. Then, the total time taken by Reduce-degree(t, m) is $O(\alpha(m) + \Lambda(m) \cdot |\mathsf{t}|)$ for some computable function $\alpha : \mathbb{N} \to \mathbb{N}$.

Reduce-height(t, m)

1. Generate $\mathcal{L}$[m]-classes and the function Colour as in the previous part.
2. Construct bottom up in t, the function Lowest-subtree : Nodes(t) $\times$ $\mathcal{L}$[m]-classes $\to$ Nodes(t) such that for any node a of t and $\delta \in \mathcal{L}$[m]-classes, Lowest-subtree(a, δ) gives a lowest (i.e. closest to a leaf) node b in $\mathsf{t}_{\geq a}$ such that Colour(b) $= \delta$. In other words, b is the only node in $\mathsf{t}_{\geq b}$ such that Colour(b) $= \delta$.
3. Let $a_1, \ldots, a_n$ be the children of root(t). Let x_i = Rainbow-subtree($\mathsf{t}_{\geq a_i}$) for $i \in \{1, \ldots, n\}$, where Rainbow-subtree(x) is described below.
4. Return $\mathsf{t}[\mathsf{t}_{\geq a_1} \mapsto \mathsf{x}_1] \ldots [\mathsf{t}_{\geq a_n} \mapsto \mathsf{x}_n]$.

Rainbow-subtree(x)

1. Let a = root(x).
2. If b = Lowest-subtree(a, Colour(a)) $\neq a$, then return Rainbow-subtree($\mathsf{x}_{\geq b}$).
3. Else, let $b_1, \ldots, b_n$ be the children of root(x). For $1 \leq i \leq n$, let y_i = Rainbow-subtree($\mathsf{x}_{\geq b_i}$).
4. Return $\mathsf{x}[\mathsf{x}_{\geq b_1} \mapsto \mathsf{y}_1] \ldots [\mathsf{x}_{\geq b_n} \mapsto \mathsf{y}_n]$.

Using similar reasoning as in the proof of Lemma 3.5(2), we can verify that algorithm Rainbow-subtree(x), that takes a subtree x of t as input, outputs a subtree y of x such that (i) $\mathsf{y} \equiv_{m,\mathcal{L}} \mathsf{x}$ and (ii) no path from the root to the leaf of y contains two distinct nodes a and b such that Colour(a) = Colour(b). Further, Rainbow-subtree(x) also satisfies the following "colour preservation" property. Let for a subtree s of t, obtained from t by removal of rooted subtrees and replacements with rooted subtrees, $\mathcal{Q}(\mathsf{s})$ be a predicate that denotes that the function Colour computed for t, when restricted to the nodes of s, is such that for any node a of s, Colour(a) gives the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{s}_{\geq a}$. Then the "colour

preservation" property says that if the input x to Rainbow-subtree satisfies $\mathcal{Q}(\cdot)$, then so does the output y of Rainbow-subtree.

From the preceding features of Rainbow-subtree, we see that the height of the output y of Rainbow-subtree(x) is at most $\Lambda(m)$. The number of "top level" recursive calls made by Rainbow-subtree(x) is linear in the degree of root(x), whereby the total time taken by Rainbow-subtree(x) is linear in $|x|$. The time taken to compute Lowest-subtree is easily seen to be $O(\Lambda(m) \cdot |t|)$. Then the time taken by Reduce-height(t, m) is $O(\eta_3(m) + \Lambda(m) \cdot |t|)$. One can verify that Reduce-height(t, m) indeed returns the desired subtree s_2 of t. ◀

4 Lifting to tree representations

We now consider the more abstract setting of tree representations of structures, and show that under suitable conditions on these representations (that a variety of classes of structures satisfy), we can lift the techniques seen in the previous section. Fix finite alphabets Σ_{int} and Σ_{leaf} (where the two alphabets are allowed to be overlapping). Let $\Sigma_{\text{rank}} \subseteq \Sigma_{\text{int}}$. Let $\rho : \Sigma_{\text{int}} \to \mathbb{N}_+$ be a fixed function. We say a class $\mathcal{T}$ of $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$-trees is *representation-feasible for* (Σ_{rank}, ρ) if $\mathcal{T}$ is closed under (label-preserving) isomorphisms, and for all trees $t = (O, \lambda) \in \mathcal{T}$ and nodes a of t, the following conditions hold:

1. Labeling condition: If a is a leaf node, resp. internal node, then the label $\lambda(a)$ belongs to Σ_{leaf}, resp. Σ_{int}.
2. Ranking by ρ: If a is an internal node and $\lambda(a)$ is in Σ_{rank}, then the number of children of a in t is exactly $\rho(\lambda(a))$.
3. Closure under rooted subtrees: The subtree $t_{\geq a}$ is in $\mathcal{T}$.
4. Closure under removal of rooted subtrees respecting Σ_{rank}: If a is an internal node, b is a child of a in t and $\lambda(a) \notin \Sigma_{\text{rank}}$, then the subtree $(t - t_{\geq b})$ is in $\mathcal{T}$.
5. Closure under replacements with rooted subtrees: If a is an internal node, then for every descendent b of a in t, the subtree $t[t_{\geq a} \mapsto t_{\geq b}]$ is in $\mathcal{T}$.

Given such a class $\mathcal{T}$ of trees as above and a class $\mathcal{S}$ of structures, let $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$ be a map that associates with each tree in $\mathcal{T}$, a structure in $\mathcal{S}$. We call Str a *representation map*. For a tree $t \in \mathcal{T}$, if $\mathfrak{A} = \mathsf{Str}(t)$, then we say t is a *tree representation* of $\mathfrak{A}$ under Str. For the purposes of our result, we consider "good" maps that would allow tree reductions of the kind seen in the previous section. We formally define these below:

▶ **Definition 4.1** ($\mathcal{L}$-good representation maps)**.** Let $\mathcal{S}$ be a class of structures and $\mathcal{T}$ be a class of $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$-trees that is representation-feasible for $(\Sigma_{\text{rank}}, \rho)$. A representation map $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$ is said to be $\mathcal{L}$-*good* if it has the following properties:

1. Isomorphism preservation: Str maps isomorphic (labeled) trees to isomorphic structures.
2. Surjectivity: Each structure in $\mathcal{S}$ has an isomorphic structure in the range of Str.
3. Monotonicity: Let $t \in \mathcal{T}$ be a tree of size ≥ 2, and a be a node of t.
   a. If $s = t_{\geq a}$, then $\mathsf{Str}(s) \hookrightarrow \mathsf{Str}(t)$.
   b. If b is a child of a in t, $\lambda(a) \notin \Sigma_{\text{rank}}$ and $z = (t - t_{\geq b})$, then $\mathsf{Str}(z) \hookrightarrow \mathsf{Str}(t)$.
   c. If b is a descendent of a in t and $z = t[t_{\geq a} \mapsto t_{\geq b}]$, then $\mathsf{Str}(z) \hookrightarrow \mathsf{Str}(t)$.
4. Composition: There exists $m_0 \in \mathbb{N}$ such that for every $m \geq m_0$ and for every $\sigma \in \Sigma_{\text{int}}$, there exists a function $f_{\sigma,m} : (\Delta_{\mathcal{S},\mathcal{L},m})^{\rho(\sigma)} \to \Delta_{\mathcal{S},\mathcal{L},m}$ if $\sigma \in \Sigma_{\text{rank}}$, and functions $f_{\sigma,m,i} : (\Delta_{\mathcal{S},\mathcal{L},m})^i \to \Delta_{\mathcal{S},\mathcal{L},m}$ for $i \in \{1, \ldots, \rho(\sigma)\}$ if $\sigma \in \Sigma_{\text{int}} \setminus \Sigma_{\text{rank}}$, with the following properties: Let $t = (O, \lambda) \in \mathcal{T}$ and a be an internal node of t such that $\lambda(a) = \sigma$, and let the children of a in t be $b_1, \ldots, b_n$. Let δ_i be the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{Str}(t_{\geq b_i})$ for $i \in \{1, \ldots, n\}$, and let δ be the $\equiv_{m,\mathcal{L}}$ class of $\mathsf{Str}(t_{\geq a})$.

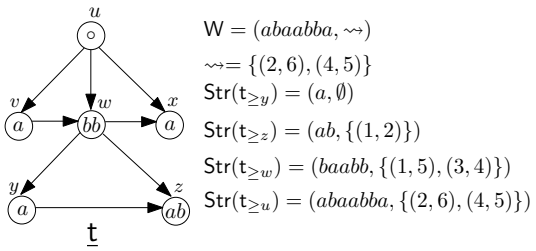


Figure 1 Nested word as a tree.

- If $\sigma \in \Sigma_{\text{rank}}$ (whereby $n = \rho(\sigma)$), then $\delta = f_{\sigma,m}(\delta_1, \ldots, \delta_n)$.
- If $\sigma \in \Sigma_{\text{int}} \setminus \Sigma_{\text{rank}}$, then δ is given as follows: Let $d = \rho(\sigma)$ and $n = r + q \cdot (d-1)$ where $1 \leq r < d$. Let $I = \{r + j \cdot (d-1) \mid 0 \leq j \leq q\}$ and for $k \in I, k \neq n$, let $\chi_{k+(d-1)} = f_{\sigma,m,d}(\chi_k, \delta_{k+1}, \ldots, \delta_{k+(d-1)})$ where $\chi_r = f_{\sigma,m,r}(\delta_1, \ldots, \delta_r)$. Then $\delta = \chi_n$.

We say $\mathcal{S}$ *admits an* $\mathcal{L}$*-good tree representation* if there exists a representation map $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$ that is $\mathcal{L}$-good. We say an $\mathcal{L}$-good representation map $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$ is *effective (resp. elementary)* if (i) $\mathcal{T}$ is recursive and (ii) there is an algorithm that, given $\mathsf{t} \in \mathcal{T}$ as input, computes $\mathsf{Str}(\mathsf{t})$ (resp. computes $\mathsf{Str}(\mathsf{t})$ in time that is bounded by an elementary function of $|\mathsf{t}|$). We now present the central result of this section, which is a lifting of Proposition 3.2 to tree representations. The proof involves an abstraction of all the ideas presented in proof of Proposition 3.2. The details of the proof can be found in [29].

▶ **Theorem 4.2.** *Let $\mathcal{S}$ be a class of structures that admits an $\mathcal{L}$-good tree representation $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$. Then the following are true:*

1. *$\mathcal{L}$-EBSP($\mathcal{S}$) holds.*
2. *If Str is effective, then there exists a computable witness function for $\mathcal{L}$-EBSP($\mathcal{S}$). Further, there exists a linear time f.p.t. algorithm for $\mathsf{MC}(\mathcal{L}, \mathcal{S})$ that decides, for every $\mathcal{L}$ sentence φ (the parameter), if a given structure $\mathfrak{A}$ in $\mathcal{S}$ satisfies φ, provided that $\mathfrak{A}$ is given in the form of a tree representation of it under Str.*
3. *If Str is elementary, then there exists an elementary witness function for $\mathcal{L}$-EBSP($\mathcal{S}$) iff the index of the $\equiv_{m,\mathcal{L}}$ relation over $\mathcal{S}$ has an elementary dependence on m.*

5 Applications to various concrete settings

A. Regular languages of words, trees and nested words. Let Σ be a finite alphabet. The notion of unordered, ordered, ranked and partially ranked Σ-trees was already introduced in Section 3.1. A Σ-tree whose underlying poset is a linear order is called a Σ-*word*. A *nested word over* Σ is a pair $(w, \rightsquigarrow)$ where w is a "usual" Σ-word and $\rightsquigarrow$ is a binary relation representing a "nested matching". Formally, if $(A, \leq)$ is the linear order underlying w, then $\rightsquigarrow$ satisfies the following: (i) for $i, j \in A$, if $i \rightsquigarrow j$, then $i \leq j$ and $i \neq j$ (ii) for $i \in A$, there is at most one $j \in A$ such that $i \rightsquigarrow j$ and at most one $l \in A$ such that $l \rightsquigarrow i$, and (iii) for $i_1, i_2, j_1, j_2 \in A$, if $i_1 \rightsquigarrow j_1$ and $i_2 \rightsquigarrow j_2$, then it is not the case that $i_1 < i_2 \leq j_1 < j_2$. (Nested words here correspond to those of [2] that have no pending calls or pending returns.)

For e.g., $\mathsf{w} = (abaabba, \{(2,6),(4,5)\})$ is a nested word over $\{a, b\}$. A nested Σ-word has a natural representation using a tree over $\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}}$, where $\Sigma_{\text{leaf}} = \Sigma \cup (\Sigma \times \Sigma)$, and $\Sigma_{\text{int}} = (\Sigma \times \Sigma) \cup \{\circ\}$. Figure 1 shows such a tree t for w. Conversely, every ordered tree over $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$, whose leaf and internal node labels belong to Σ_{leaf} and Σ_{int} resp., represents a nested Σ-word.

The notion of *regular languages* of words, trees (of all the aforementioned kinds) and nested words, and its correspondence with MSO definability, are well-studied [5, 2]. We say a class of words, trees or nested words is regular if it is the class of models of an MSO sentence.

▶ **Theorem 5.1.** *Given finite alphabets Σ, Ω such that $\Omega \subseteq \Sigma$, and a function $\rho : \Omega \to \mathbb{N}$, let* $\mathsf{Words}(\Sigma)$, $\mathsf{Unordered\text{-}trees}(\Sigma)$, $\mathsf{Ordered\text{-}trees}(\Sigma)$, $\mathsf{Partially\text{-}ranked\text{-}trees}(\Sigma, \Omega, \rho)$ *and* $\mathsf{Nested\text{-}words}(\Sigma)$ *denote resp. the classes of all Σ-words, all unordered Σ-trees, all ordered Σ-trees, all ordered Σ-trees partially ranked by (Ω, ρ), and all nested Σ-words. Let $\mathcal{S}$ be a regular subclass of any of these classes. Then $\mathcal{L}$-$\mathsf{EBSP}(\mathcal{S})$ holds with a computable witness function. Further, any witness function for $\mathcal{L}$-$\mathsf{EBSP}(\mathcal{S})$ is necessarily non-elementary.*

Proof Idea. We first show MSO-$\mathsf{EBSP}(\mathcal{S})$ holds when $\mathcal{S}$ is exactly one of the classes mentioned in the statement of the theorem. That $\mathcal{L}$-$\mathsf{EBSP}(\cdot)$ holds for a regular subclass follows, because (i) MSO-$\mathsf{EBSP}(\cdot)$ is preserved under MSO definable subclasses, and (ii) MSO-$\mathsf{EBSP}(\cdot)$ implies FO-$\mathsf{EBSP}(\cdot)$. Consider $\mathcal{S} = \mathsf{Unordered\text{-}trees}(\Sigma)$ (the other cases of trees have been covered by Proposition 3.2). Let $\mathcal{T}_1$ be the class of all ordered $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$-trees where $\Sigma_{\text{int}} = \Sigma_{\text{leaf}} = \Sigma$; then $\mathcal{T}_1$ is representation-feasible for $(\Sigma_{\text{rank}}, \rho)$ where $\Sigma_{\text{rank}} = \emptyset$ and ρ is the constant function of value 2. There is now a natural representation map $\mathsf{Str}_1 : \mathcal{T}_1 \to \mathcal{S}$ such that Str_1 "forgets" the ordering among the children of any node of its input tree. Using an MSO composition lemma for unordered trees (see Appendix A), we can see that Str_1 is an elementary MSO-good representation map, whereby using Theorem 4.2, we are done. Consider $\mathcal{S} = \mathsf{Nested\text{-}words}(\Sigma)$. For $\Sigma_{\text{leaf}} = \Sigma \cup (\Sigma \times \Sigma)$ and $\Sigma_{\text{int}} = (\Sigma \times \Sigma) \cup \{\circ\}$, if $\mathcal{T}_2$ is the class of all ordered $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$-trees whose leaf labels belong to Σ_{leaf} and internal node labels belong to Σ_{int}, then $\mathcal{T}_2$ is representation-feasible for $(\Sigma_{\text{rank}}, \rho)$ where again $\Sigma_{\text{rank}} = \emptyset$ and ρ is the constant function 2. There is then a natural map $\mathsf{Str}_2 : \mathcal{T}_2 \to \mathcal{S}$ as described for the example above. By an MSO composition lemma for nested words (see Appendix A), it follows that Str_2 is an elementary MSO-good representation map. We are then done by Theorem 4.2 again. The non-elementariness of witness functions is due to Theorem 4.2 and the non-elementary dependence on m, of the index of the $\equiv_{m,\mathcal{L}}$ relation over words [10]. ◀

B. n-partite cographs. The class of n-partite cographs, introduced in [12], can be defined up to isomorphism as the range of the map Str described as follows. Let $\Sigma_{\text{leaf}} = [n] = \{1, \ldots, n\}$ and $\Sigma_{\text{int}} = \{f \mid f : [n] \times [n] \to \{0, 1\}\}$. Let $\mathcal{T}$ be the class of all ordered $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$-trees whose leaf labels belong to Σ_{leaf} and internal node labels belong to Σ_{int}. Then $\mathcal{T}$ is representation-feasible for $(\Sigma_{\text{rank}}, \rho)$ where $\Sigma_{\text{rank}} = \emptyset$ and $\rho : \Sigma_{\text{int}} \to \mathbb{N}_+$ is the constant function 2. Consider $\mathsf{Str} : \mathcal{T} \to \mathsf{Graphs}$ defined as follows, where Graphs is the class of all undirected graphs: For $\mathsf{t} = (O, \lambda) \in \mathcal{T}$ where $O = ((A, \leq), \lessdot)$ is an ordered unlabeled tree and λ is the labeling function, $\mathsf{Str}(\mathsf{t}) = G = (V, E)$ is such that (i) V is exactly the set of leaf nodes of t (ii) for $a, b \in V$, if $c = a \wedge b$ is the greatest common ancestor (under $\leq$) of a and b in t, then $\{a, b\} \in E$ iff $\lambda(c)(\lambda(a), \lambda(b)) = 1$. We now have the following result. Below, a Σ-*labeled* n-partite cograph is a pair (G, ν) where G is an n-partite cograph and $\nu : V \to \Sigma$ is a labeling function. Also, "hereditary" means "closed under substructures".

▶ **Theorem 5.2.** *Given $n \in \mathbb{N}$ and a finite alphabet Σ, let* $\mathsf{Labeled\text{-}n\text{-}partite\text{-}cographs}(\Sigma)$ *be the class of all Σ-labeled n-partite cographs. Let $\mathcal{S}$ be any hereditary subclass of this class. Then $\mathcal{L}$-$\mathsf{EBSP}(\mathcal{S})$ holds with a computable witness function. Whereby, each of the graph classes below satisfies $\mathcal{L}$-$\mathsf{EBSP}(\cdot)$ with a computable witness function. Further, the classes with bounded parameters as mentioned below have elementary functions witnessing $\mathcal{L}$-$\mathsf{EBSP}(\cdot)$.*

1. *Any hereditary class of n-partite cographs, for each $n \in \mathbb{N}$.*
2. *Any hereditary class of graphs of bounded shrub-depth.*

3. *Any hereditary class of graphs of bounded $\mathcal{SC}$-depth.*
4. *Any hereditary class of graphs of bounded tree-depth.*
5. *Any hereditary class of cographs.*

Proof Idea. We first show the result for $\mathcal{S} = \mathsf{Labeled\text{-}n\text{-}partite\text{-}cographs}(\Sigma)$. The result for the various specific classes mentioned in the statement follows from the fact that $\mathcal{L}$-EBSP$(\cdot)$ is closed under hereditary subclasses, and that all of the specific classes are hereditary subclasses of n-partite cographs [12]. Let $\Sigma_{\text{leaf}} = [n] \times \Sigma$ and $\Sigma_{\text{int}} = \{f \mid f : [n] \times [n] \to \{0,1\}\}$. Then consider the class $\mathcal{T}$ of ordered $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$-trees and the representation map $\mathsf{Str} : \mathcal{T} \to \mathsf{Labeled\text{-}n\text{-}partite\text{-}cographs}(\Sigma)$ exactly of the respective kinds described above for n-partite cographs. By an $\mathcal{L}$-composition lemma for labeled n-partite cographs (see Appendix A), we see that Str is elementary and $\mathcal{L}$-good, whereby we are done by Theorem 4.2. That the graph classes with the bounded parameters above have elementary witness functions follows again from Theorem 4.2 and elementary dependence on m, of the index of the $\equiv_{m,\mathcal{L}}$ relation over these classes (the latter follows from Theorem 3.2 of [11]). ◀

C. Classes generated using translation schemes. We look operations on classes of structures, that are "implementable" using quantifier-free translation schemes [23]. Given a vocabulary τ, let $\tau_{\text{disj-un},n}$ be the vocabulary obtained by expanding τ with n fresh unary predicates $P_1, \ldots, P_n$. Given τ-structures $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$ (assumed disjoint w.l.o.g.), the *n-disjoint sum of* $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$, denoted $\bigoplus_{i=1}^{i=n} \mathfrak{A}_i$, is the $\tau_{\text{disj-un},n}$-structure obtained upto isomorphism, by expanding the disjoint union $\bigsqcup_{i=1}^{i=n} \mathfrak{A}_i$ with $P_1, \ldots, P_n$ interpreted respectively as the universe of $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$. Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be given classes of structures. A quantifier-free $(t, \tau_{\text{disj-un},n}, \tau, \text{FO})$-translation scheme Ξ gives rise to an n-ary operation $\mathsf{O} : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \{\Xi(\bigoplus_{i=1}^{i=n} \mathfrak{A}_i) \mid \mathfrak{A}_i \in \mathcal{S}_i, 1 \leq i \leq n\}$ defined as $\mathsf{O}_1(\mathfrak{A}_1, \ldots, \mathfrak{A}_n) = \Xi(\bigoplus_{i=1}^{i=n} \mathfrak{A}_i)$. In this case, we say that O is *implementable using* Ξ. We say an operation is *quantifier-free*, if it is of the kind O just described. For a quantifier-free operation O, define the *dimension* of O to be the minimum of the dimensions of the quantifier-free translation schemes that implement O. We say O is "sum-like" if its dimension is one, else we say O is "product-like". We say O is *monotone* if any input of O is embeddable in the output of O. We say O *obeys $\mathcal{L}$-composition* if for each m, whenever an input of O is replaced with an $\mathcal{L}[m]$-equivalent input, the output of O is replaced with an $\mathcal{L}[m]$-equivalent output. The well-studied unary graph operations like complementation, transpose, and the line-graph operation, and binary operations like disjoint union and join are all sum-like, monotone and obey $\mathcal{L}$-composition. Likewise, the well-studied Cartesian, tensor, lexicographic, and strong products are all product-like, monotone and obey $\mathcal{L}$-composition. The central result of this section is as stated below. A proof sketch for this result in presented in Appendix B.

▶ **Theorem 5.3.** *Let $\mathcal{S}_1, \ldots, \mathcal{S}_n, \mathcal{S}$ be classes of structures and let $\mathsf{O} : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathcal{S}$ be a surjective n-ary quantifier-free operation. Then the following are true:*

1. *For each of the following cases, if $\mathcal{L}$-EBSP$(\mathcal{S}_i)$ holds (with computable/elementary witness functions) for each $i \in \{1, \ldots, n\}$, then $\mathcal{L}$-EBSP$(\mathcal{S})$ holds as well (with computable/elementary witness functions):* (i) O *is sum-like* (ii) O *is product-like and $\mathcal{L} = FO$.*
2. *Suppose $\mathcal{S}_i$ admits an effective $\mathcal{L}$-good tree representation for each $i \in \{1, \ldots, n\}$, and O is monotone and obeys $\mathcal{L}$-composition. Then there exists an effective $\mathcal{L}$-good tree representation $\mathsf{Str} : \mathcal{T} \to \mathcal{Z}$ for the class $\mathcal{Z} = \mathcal{S} \cup \bigcup_{i=1}^{i=n} \mathcal{S}_i$.*
3. *Let Str be as given by the previous point. Then there is a linear time f.p.t. algorithm for $\mathsf{MC}(\mathcal{L}, \mathcal{S})$ that decides, for every $\mathcal{L}$ sentence φ (the parameter), if a given structure $\mathfrak{A}$ in $\mathcal{S}$ satisfies φ, provided that $\mathfrak{A}$ is given in the form of a tree representation of it under Str.*

▶ Discussion. Theorems 5.1, 5.2, 5.3 and 4.2 jointly show that the various posets and graph classes described in this section admit linear time f.p.t. algorithms for $\mathsf{MC}(\mathcal{L}, \cdot)$, provided an $\mathcal{L}$-good tree representation of the input structure is given. For structures coming from the classes of words, the various kinds of trees, nested words and cographs, we can indeed even construct, in linear time, the Hasse diagram of an $\mathcal{L}$-good tree representation for the structure, given its standard presentation (this is easy to see for the first three kinds of classes; for the case of cographs, see [18]). It turns out that the techniques used in our f.p.t. algorithms can be easily adapted to work even when the input structures are presented using the aforementioned diagrams. This then enables getting (unconditional) linear time f.p.t. algorithms for $\mathsf{MC}(\mathcal{L}, \cdot)$ for the cases of words, trees, nested words and cographs, thereby matching known f.p.t. results for $\mathsf{MC}(\mathcal{L}, \cdot)$ concerning these classes [10, 2, 19]. Going further, to the best of our knowledge, the f.p.t. results for n-partite cographs and those for classes generated using trees of quantifier-free operations, that are entailed by Theorems 5.2, 5.3 and 4.2, are new. Our proofs can then be seen as giving a different and unified technique to show existing f.p.t. results, in addition to giving new results. We mention however that if the dependence on the parameter in our f.p.t. algorithms is also considered, then our results (which give only computable parameter dependence) are weaker than those in [11] which show that for classes of bounded tree-depth/$\mathcal{SC}$-depth/shrub-depth, there are linear time f.p.t. algorithms for $\mathcal{L}$ model checking, that have elementary parameter dependence.

6 Logical fractals

We define a strengthening of $\mathcal{L}$-EBSP that asserts "logical self-similarity" at "all scales" for a suitable notion of scale. Towards the formal definition, call a function $f : \mathbb{N}_+ \to \mathbb{N}_+$ a *scale function* if it is strictly increasing. The i^{th} *scale*, denoted $\langle i \rangle_f$, is defined as the interval $[f(i-1)+1, f(i)] = \{j \mid f(i-1)+1 \leq j \leq f(i)\}$ for $i > 1$, and $[1, f(1)] = \{j \mid 1 \leq j \leq f(1)\}$.

▶ **Definition 6.1** (Logical fractal)**.** Given a class $\mathcal{S}$ of structures, we say $\mathcal{S}$ is an $\mathcal{L}$-*fractal*, if there exists a function $\theta_{(\mathcal{S},\mathcal{L})} : \mathbb{N}_+^2 \to \mathbb{N}_+$ such that (i) $\theta_{(\mathcal{S},\mathcal{L})}(m)$ is a scale function for all $m \in \mathbb{N}$, and (ii) for each structure $\mathfrak{A}$ of $\mathcal{S}$ and each $m \in \mathbb{N}$, if f is the function $\theta_{(\mathcal{S},\mathcal{L})}(m)$ and $|\mathfrak{A}| \in \langle i \rangle_f$ for some $i \in \mathbb{N}$, then for all $j < i$, there exists a substructure $\mathfrak{B}$ of $\mathfrak{A}$ in $\mathcal{S}$, such that $|\mathfrak{B}| \in \langle j \rangle_f$ and $\mathfrak{B} \equiv_{m,\mathcal{L}} \mathfrak{A}$. We say $\theta_{(\mathcal{S},\mathcal{L})}$ is a *witness* to the $\mathcal{L}$-fractal property of $\mathcal{S}$.

Call a representation map $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$ as $\mathcal{L}$-*great* if (i) it is $\mathcal{L}$-good, and (ii) there is a strictly increasing function $\beta : \mathbb{N} \to \mathbb{N}$ such that for every $\mathsf{t}, \mathsf{s} \in \mathcal{T}$, if $|(|\mathsf{t}| - |\mathsf{s}|)| \leq n$, then $|(|\mathsf{Str}(\mathsf{t})| - |\mathsf{Str}(\mathsf{s})|)| \leq \beta(n)$. In such a case, we say $\mathcal{S}$ *admits* an $\mathcal{L}$-great tree representation. We now have the following result. We present a proof sketch in Appendix C.

▶ **Proposition 6.2.** *If $\mathcal{S}$ admits an $\mathcal{L}$-great tree representation, then $\mathcal{S}$ is an $\mathcal{L}$-fractal.*

One can easily verify that for all the examples of posets and graphs considered in Section 5, their $\mathcal{L}$-good representation maps as described in this section are indeed $\mathcal{L}$-great too, whereby all these classes are logical fractals. Further, the logical fractal property is preserved under all examples of operations on structures seen in Section 5; see Appendix C for a proof sketch.

7 Conclusion

We presented a natural finitary analogue of the well-studied downward Löwenheim-Skolem property from classical model theory, denoted $\mathcal{L}$-EBSP, and showed that this property is enjoyed by various classes of interest in computer science, whereby all these classes can

be seen to admit a natural finitary version of the downward Löwenheim-Skolem theorem. The aforesaid classes admit tree representations for their structures, using which we obtain linear time f.p.t. algorithms for FO and MSO model checking for these classes (when the structures in the classes are presented using their tree representations). Finally, the aforesaid classes possess a fractal like property, one based on logic. These observations open up several interesting and challenging directions for future work, some of which we mention below.

1. Under what conditions on a class of structures is the index of the $\equiv_{m,\mathcal{L}}$ relation over the class an elementary function of m? Investigating this question for classes admitting elementary $\mathcal{L}$-good tree representations might yield insights for linear time f.p.t. algorithms for $\mathcal{L}$ model checking over these classes, that have elementary parameter dependence.
2. We have adopted the composition method for proving f.p.t. results for the classes we have considered. However, automata based methods for f.p.t. results have also been well-studied in the literature [6, 14, 11]. Since all our classes have representations using trees, a natural question is whether we can give tree automata based proofs for them, and more generally for Theorem 4.2. Given the existence of pumping lemmas for trees, the aforementioned proofs would also, intuitively speaking, lend themselves to investigating the *upward* Löwenheim-Skolem property for all of our classes.
3. The requirement of a small and logically similar *substructure* in the $\mathcal{L}$-EBSP definition, causes some well-studied classes of structures to not satisfy $\mathcal{L}$-EBSP. For instance, the class of all graphs of tree-width $\leq k$ does not satisfy $\mathcal{L}$-EBSP for any k. This motivates asking whether these classes satisfy variants of $\mathcal{L}$-EBSP in which "substructure" is replaced with other natural relations, and if they do, then whether the witness functions are computable. As a step in this direction, we show that the variant of $\mathcal{L}$-EBSP in which "substructure" is replaced with "minor", is satisfied by the class of all graphs, and every minor-hereditary subclass of it. Any witness function however, turns out to be non-recursive in general.
4. All our examples of $\mathcal{L}$-EBSP classes (resp. logical fractals) are defined using structural conditions. This leads us to asking the converse, and hence the following: Is there a structural characterization of $\mathcal{L}$-EBSP (resp. of logical fractals)? We believe that an answer to this question, even under reasonable assumptions, would yield new classes that are well-behaved from both the logical and the algorithmic perspectives.

Acknowledgements. I express my deepest gratitude to Bharat Adsul for various insightful discussions and critical feedback that have helped in preparing this paper. I also thank the anonymous referees of the paper, and of an earlier version of it, for their valuable comments.

References

1 Natasha Alechina and Yuri Gurevich. Syntax vs. semantics on finite structures. In *Structures in Logic and Computer Science. A Selection of Essays in Honor of A. Ehrenfeucht*, pages 14–33. Springer-Verlag, 1997.

2 Rajeev Alur and Parthasarathy Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3), 2009.

3 Albert Atserias, Anuj Dawar, and Martin Grohe. Preservation under extensions on well-behaved finite structures. *SIAM J. Comput.*, 38(4):1364–1381, 2008. doi:10.1137/060658709.

4 Michael Barnsley. *Fractals Everywhere*. Academic Press Professional, Inc., 1988.

5 Hubert Comon, Max Dauchet, Remi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications, 2007. release October 12, 2007. URL: http://www.grappa.univ-lille3.fr/tata.

6 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

7 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

8 Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. In *LICS 2012, Croatia, June 25-28, 2012*, pages 265–274, 2012.

9 Solomon Feferman and Robert Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47(1):57–103, 1959.

10 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004.

11 Jakub Gajarsky and Petr Hlinený. Kernelizing MSO properties of trees of fixed height, and some consequences. *Log. Meth. Comp. Sci.*, 11(19):1–26, 2015.

12 Robert Ganian, Petr Hlinený, Jaroslav Nešetřil, Jan Obdrzálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When trees grow low: Shrubs and fast MSO1. In *MFCS 2012, Bratislava, Slovakia, August 27-31, 2012*, pages 419–430, 2012.

13 Martin Grohe. Some remarks on finite Löwenheim-Skolem theorems. *Math. Log. Q.*, 42:569–571, 1996.

14 Martin Grohe. Logic, graphs, and algorithms. *Logic and automata*, 2:357–422, 2008.

15 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. *Model Theoretic Methods in Finite Combinatorics*, 558:181–206, 2011.

16 Yuri Gurevich. Toward logic tailored for computational complexity. In Michael M. Richter et al., editor, *Computation and Proof Theory: Proceedings of the Logic Colloquium held in Aachen, July 18-23, 1983, Part II*, pages 175–216. Springer-Verlag, 1984.

17 Frederik Harwath, Lucas Heimberg, and Nicole Schweikardt. Preservation and decomposition theorems for bounded degree structures. *Log. Meth. Comp. Sci.*, 11(4), 2015.

18 Beverly Jamison and Stephan Olariu. Recognizing P_4-sparse graphs in linear time. *SIAM Journal on Computing*, 21(2):381–406, 1992.

19 Beverly Jamison and Stephan Olariu. Linear time optimization algorithms for P_4-sparse graphs. *Discrete Applied Mathematics*, 61(2):155–175, 1995.

20 Leonid Libkin. *Elements of Finite Model Theory*. Springer-Verlag, 2004.

21 Per Lindström. A characterization of elementary logic. In Sören Halldén, editor, *Modality, Morality and Other Problems of Sense and Nonsense*, pages 189–191. Lund,Gleerup, 1973.

22 Leopold Löwenheim. Über möglichkeiten im relativkalkül. *Mathematische Annalen*, 76(4):447–470, 1915.

23 Johann A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Ann. Pure Appl. Logic*, 126(1-3):159–213, 2004.

24 Anatoly I. Maltsev. Untersuchungen aus dem Gebiete der mathematischen Logik. *Matematicheskii Sbornik*, n.s.(1):323–336, 1936.

25 Jaroslav Nešetřil and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006.

26 Eric Rosen. Some aspects of model theory and finite structures. *Bull. Symbolic Logic*, 8(3):380–403, 2002.

27 Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):15:1–15:53, 2008.

28 Abhisekh Sankaran. A generalization of the Łoś-Tarski preservation theorem. *CoRR*, abs/1609.06297, 2016. URL: `http://arxiv.org/abs/1609.06297`.

29 Abhisekh Sankaran. A finitary analogue of the downward Löwenheim-Skolem property. *CoRR*, abs/1705.04493, 2017. URL: `http://arxiv.org/abs/1705.04493`.

30 Abhisekh Sankaran, Bharat Adsul, and Supratik Chakraborty. A generalization of the Łoś-Tarski preservation theorem over classes of finite structures. In *MFCS 2014, Budapest, Hungary, August 25-29, 2014, Part I*, pages 474–485, 2014.
31 Abhisekh Sankaran, Bharat Adsul, and Supratik Chakraborty. A generalization of the Łoś-Tarski preservation theorem. *Ann. Pure Appl. Logic*, 167(3):189–210, 2016.
32 J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, 2(1):57–81, 1968.
33 Wolfgang Thomas. *Ehrenfeucht games, the composition method, and the monadic theory of ordinal words*, pages 118–143. Springer Berlin Heidelberg, 1997.
34 Jouko Väänänen. Pseudo-finite model theory. *Mat. Contemp*, 24(8th):169–183, 2003.

A Feferman-Vaught style composition lemmas

We present Feferman-Vaught style composition lemmas for various classes of structures. While these lemmas for ordered and unordered trees, as presented here, is possibly known, to the best of our knowledge these lemmas for nested words and n-partite cographs (again as presented here) are new. The proofs of all these lemmas can be found in [29].

A. Ordered trees. To state the composition lemma, we first introduce some terminology. For a finite alphabet Ω, given ordered Ω-trees t, s having disjoint sets of nodes (w.l.o.g.) and a non-root node a of t, the *join of* s *to* t *to the right of* a, denoted $\mathsf{t} \cdot_a^{\rightarrow} \mathsf{s}$, is defined as the tree obtained by making s as a new child subtree of the parent of a in t, at the successor position of the position of a among the children of the parent of a in t. We can similarly define the *join of* s *to* t *to the left of* a, denoted $\mathsf{t} \cdot_a^{\leftarrow} \mathsf{s}$. Likewise, for t and s as above, if a is a leaf node of t, we can define the *join of* s *to* t *below* a, denoted $\mathsf{t} \cdot_a^{\uparrow} \mathsf{s}$, as the tree obtained up to isomorphism by making the root of s as a child of a.

▶ **Lemma A.1** (Composition lemma for ordered trees). *For a finite alphabet Ω, let $\mathsf{t}_i, \mathsf{s}_i$ be non-empty ordered Ω-trees, and let a_i be a non-root node of t_i, for $i \in \{1,2\}$. Let $m \geq 2$ and suppose that $(\mathsf{t}_1, a_1) \equiv_{m,\mathcal{L}} (\mathsf{t}_2, a_2)$ and $\mathsf{s}_1 \equiv_{m,\mathcal{L}} \mathsf{s}_2$. Then each of the following hold.*
1. $((\mathsf{t}_1 \cdot_{a_1}^{\rightarrow} \mathsf{s}_1), a_1) \equiv_{m,\mathcal{L}} ((\mathsf{t}_2 \cdot_{a_2}^{\rightarrow} \mathsf{s}_2), a_2)$
2. $((\mathsf{t}_1 \cdot_{a_1}^{\leftarrow} \mathsf{s}_1), a_1) \equiv_{m,\mathcal{L}} ((\mathsf{t}_2 \cdot_{a_2}^{\leftarrow} \mathsf{s}_2), a_2)$
3. $((\mathsf{t}_1 \cdot_{a_1}^{\uparrow} \mathsf{s}_1), a_1) \equiv_{m,\mathcal{L}} ((\mathsf{t}_2 \cdot_{a_2}^{\uparrow} \mathsf{s}_2), a_2)$ *if a_1, a_2 are leaf nodes of $\mathsf{t}_1, \mathsf{t}_2$ resp.*

B. Unordered trees. We introduce terminology akin to that introduced for ordered trees above. Given unordered trees t and s, and a node a of t, define the *join of* s *to* t *at* a, denoted $\mathsf{t} \cdot_a \mathsf{s}$, as follows: Let s' be an isomorphic copy of s whose set of nodes is disjoint with the set of nodes of t. Then $\mathsf{t} \cdot_a \mathsf{s}$ is defined up to isomorphism as the tree obtained by making s' as a new child subtree of a in t.

▶ **Lemma A.2** (Composition lemma for unordered trees). *For a finite alphabet Ω, let $\mathsf{t}_i, \mathsf{s}_i$ be non-empty unordered Ω-trees, and let a_i be a node of t_i, for $i \in \{1,2\}$. For $m \in \mathbb{N}$, suppose that $(\mathsf{t}_1, a_1) \equiv_{m,\mathcal{L}} (\mathsf{t}_2, a_2)$ and $\mathsf{s}_1 \equiv_{m,\mathcal{L}} \mathsf{s}_2$. Then $((\mathsf{t}_1 \cdot_{a_1} \mathsf{s}_1), a_1) \equiv_{m,\mathcal{L}} ((\mathsf{t}_2 \cdot_{a_2} \mathsf{s}_2), a_2)$.*

C. Nested words. We first define the notion of *insert of a nested word* v *in a nested word* u *at a given position* e *of* u.

▶ **Definition A.3** (Insert). Let $\mathsf{u} = (A_\mathsf{u}, \leq_\mathsf{u}, \lambda_\mathsf{u}, \rightsquigarrow_\mathsf{u})$ and $\mathsf{v} = (A_\mathsf{v}, \leq_\mathsf{v}, \lambda_\mathsf{v}, \rightsquigarrow_\mathsf{v})$ be given nested Σ-words, and let e be a position in u. The *insert of* v *in* u *at* e, denoted $\mathsf{u} \uparrow_e \mathsf{v}$, is a nested Σ-word defined as below.

1. If u and v have disjoint sets of positions, then $\mathsf{u} \uparrow_e \mathsf{v} = (A, \leq, \lambda, \rightsquigarrow)$ where
   - $A = A_{\mathsf{u}} \sqcup A_{\mathsf{v}}$
   - $\leq = \leq_{\mathsf{u}} \cup \leq_{\mathsf{v}} \cup \{(i,j) \mid i \in A_{\mathsf{u}}, j \in A_{\mathsf{v}}, i \leq_{\mathsf{u}} e\} \cup \{(j,i) \mid i \in A_{\mathsf{u}}, j \in A_{\mathsf{v}}, e \leq_{\mathsf{u}} i, e \neq i\}$
   - $\lambda(a) = \lambda_{\mathsf{u}}(a)$ if $a \in A_{\mathsf{u}}$, else $\lambda(a) = \lambda_{\mathsf{v}}(a)$
   - $\rightsquigarrow = \rightsquigarrow_{\mathsf{u}} \cup \rightsquigarrow_{\mathsf{v}}$
2. If u and v have overlapping sets of positions, then let v_1 be an isomorphic copy of v whose set of positions is disjoint with that of u. Then $\mathsf{u} \uparrow_e \mathsf{v}$ is defined up to isomorphism as $\mathsf{u} \uparrow_e \mathsf{v}_1$.

In the special case that e is the last (under $\leq_{\mathsf{u}}$) position of u, we denote $\mathsf{u} \uparrow_e \mathsf{v}$ as $\mathsf{u} \cdot \mathsf{v}$, and call the latter as the *concatenation of* v *with* u.

▶ **Lemma A.4** (Composition lemma for nested words). *For a finite alphabet Σ, let $\mathsf{u}_i, \mathsf{v}_i \in$ Nested-words(Σ), and let e_i be a position in u_i for $i \in \{1,2\}$. Then the following hold for each $m \in \mathbb{N}$.*

1. *If $(\mathsf{u}_1, e_1) \equiv_{m,\mathcal{L}} (\mathsf{u}_2, e_2)$ and $\mathsf{v}_1 \equiv_{m,\mathcal{L}} \mathsf{v}_2$, then $(\mathsf{u}_1 \uparrow_{e_1} \mathsf{v}_1) \equiv_{m,\mathcal{L}} (\mathsf{u}_2 \uparrow_{e_2} \mathsf{v}_2)$.*
2. *$\mathsf{u}_1 \equiv_{m,\mathcal{L}} \mathsf{u}_2$ and $\mathsf{v}_1 \equiv_{m,\mathcal{L}} \mathsf{v}_2$, then $\mathsf{u}_1 \cdot \mathsf{v}_1 \equiv_{m,\mathcal{L}} \mathsf{u}_2 \cdot \mathsf{v}_2$.*

D. n-partite cographs. Let $\mathcal{T}$ and Str be as described in the proof idea of Theorem 5.2. Specifically, $\mathcal{T}$ is the class of all $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$-trees whose leaf labels belong to Σ_{leaf} and internal node labels belong to Σ_{int}, where $\Sigma_{\text{leaf}} = [n] \times \Sigma$ and $\Sigma_{\text{int}} = \{f \mid f : [n] \times [n] \to \{0,1\}\}$. The representation map $\mathsf{Str} : \mathcal{T} \to$ Labeled-n-partite-cographs(Σ) is exactly of the kind described for n-partite cographs, that maps a tree in $\mathcal{T}$ to the labeled n-partite graph that it represents.

▶ **Lemma A.5** (Composition lemma for n-partite cographs). *For $i \in \{1,2\}$, let $(G_i, \nu_{i,1})$ and $(H_i, \nu_{i,2})$ be graphs in* Labeled-n-partite-cographs(Σ). *Suppose t_i and s_i are trees of $\mathcal{T}$ such that $\mathsf{Str}(\mathsf{t}_i) = (G_i, \nu_{i,1})$, $\mathsf{Str}(\mathsf{s}_i) = (H_i, \nu_{i,2})$, and the labels of $\mathsf{root}(\mathsf{t}_i)$ and $\mathsf{root}(\mathsf{s}_i)$ are the same. Let $\mathsf{z}_i = \mathsf{t}_i \odot \mathsf{s}_i$ and $\mathsf{Str}(\mathsf{z}_i) = (Z_i, \nu_i)$ for $i \in \{1,2\}$. For each $m \in \mathbb{N}$, if $(G_1, \nu_{1,1}) \equiv_{m,\mathcal{L}} (G_2, \nu_{2,1})$ and $(H_1, \nu_{1,2}) \equiv_{m,\mathcal{L}} (H_2, \nu_{2,2})$, then $(Z_1, \nu_1) \equiv_{m,\mathcal{L}} (Z_2, \nu_2)$.*

B Proof sketch for Theorem 5.3

Recall from Section 2 that a $(t, \tau, \nu, \mathcal{L})$-translation scheme Ξ defines a map from τ-structures to ν-structures, that we denote by Ξ again. For a class $\mathcal{S}$ of τ-structures, we let $\Xi(\mathcal{S})$ denote the class $\{\Xi(\mathfrak{A}) \mid \mathfrak{A} \in \mathcal{S}\}$. Let n-disjoint-sum$(\mathcal{S}_1, \ldots, \mathcal{S}_n) = \{\bigoplus_{i=1}^{i=n} \mathfrak{A}_i \mid \mathfrak{A}_i \in \mathcal{S}_i, 1 \leq i \leq n\}$, where $\bigoplus_{i=1}^{i=n} \mathfrak{A}_i$ denotes the n-disjoint sum of $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$. Call a quantifier-free translation scheme *scalar* if its dimension is one. We now have the following three results that enable us to prove Theorem 5.3. The proofs of these results can be found in [29].

▶ **Lemma B.1.** *Let $\mathcal{S}$ be a class of τ-structures and Ξ be a quantifier-free (t, τ, ν, FO)-translation scheme. Given structures $\mathfrak{A}$ and $\mathfrak{B}$ from $\mathcal{S}$, if $\mathfrak{B} \subseteq \mathfrak{A}$, then $\Xi(\mathfrak{B}) \subseteq \Xi(\mathfrak{A})$.*

▶ **Lemma B.2.** *Let $\mathcal{S}, \mathcal{S}_1, \ldots, \mathcal{S}_n$ be classes of structures for $n \geq 1$. If $\mathcal{L}$-EBSP$(\mathcal{S}_i)$ is true for each $i \in \{1, \ldots, n\}$, then so is $\mathcal{L}$-EBSP$(n$-disjoint-sum$(\mathcal{S}_1, \ldots, \mathcal{S}_n))$. Further, if there is a computable/elementary witness function for $\mathcal{L}$-EBSP$(\mathcal{S}_i)$ for each $i \in \{1, \ldots, n\}$, then there is a computable/elementary witness function for $\mathcal{L}$-EBSP$(n$-disjoint-sum$(\mathcal{S}_1, \ldots, \mathcal{S}_n))$ as well.*

▶ **Proposition B.3.** *Let $\mathcal{S}$ be class of τ-structures and Ξ be a quantifier-free (t, τ, ν, FO)-translation scheme. Then the following are true:*

1. *If FO-EBSP$(\mathcal{S})$ is true, then so is FO-EBSP$(\Xi(\mathcal{S}))$.*
2. *If Ξ is scalar and MSO-EBSP$(\mathcal{S})$ is true, then so is MSO-EBSP$(\Xi(\mathcal{S}))$.*

In each of the implications above, a computable/elementary witness function for the antecedent implies a computable/elementary witness function for the consequent.

Proof of Theorem 5.3.

(1) Follows easily from Lemma B.2 and Proposition B.3.

(2) Let $\mathsf{Str}_i : \mathcal{T}_i \to \mathcal{S}_i$ be an effective $\mathcal{L}$-good representation map for $1 \leq i \leq n$, where $\mathcal{T}_i$ is a class of trees over $(\Sigma^i_{\text{int}} \cup \Sigma^i_{\text{leaf}})$ that is representation feasible for $(\Sigma^i_{\text{rank}}, \rho_i)$.
Let O be a new label that is not in $(\Sigma^i_{\text{int}} \cup \Sigma^i_{\text{leaf}})$ for any $i \in \{1, \ldots, n\}$. Define $\Sigma_{\text{int}}, \Sigma_{\text{leaf}}, \Sigma_{\text{rank}}$ and $\rho : \Sigma_{\text{int}} \to \mathbb{N}_+$ as follows:
- $\Sigma_{\text{int}} = \{O\} \cup \bigcup_{i=1}^{i=n} \Sigma^i_{\text{int}}$
- $\Sigma_{\text{leaf}} = \bigcup_{i=1}^{i=n} \Sigma^i_{\text{leaf}}$
- $\Sigma_{\text{rank}} = \{O\} \cup \bigcup_{i=1}^{i=n} \Sigma^i_{\text{rank}}$
- $\rho = \{(O, n)\} \cup \bigcup_{i=1}^{i=n} \rho_i$.

Let $\widehat{\mathcal{T}}$ be the class of all trees over $(\Sigma_{\text{int}} \cup \Sigma_{\text{leaf}})$ obtained by taking $\mathsf{t}_i \in \mathcal{T}_i$ for $1 \leq i \leq n$, and making $\mathsf{t}_1, \ldots, \mathsf{t}_n$ as child subtrees (and in that order) of a new root node whose label is O. Let $\mathcal{T} = \widehat{\mathcal{T}} \cup \bigcup_{i=1}^{i=n} \mathcal{T}_i$. Verify that $\mathcal{T}$ is indeed representation feasible for $(\Sigma_{\text{rank}}, \rho)$. Let $\mathsf{Str} : \mathcal{T} \to \mathcal{Z}$ be such that for $\mathsf{t} \in \mathcal{T}$, if $\mathsf{t} \in \mathcal{T}_i$, then $\mathsf{Str}(\mathsf{t}) = \mathsf{Str}_i(\mathsf{t})$. Else, let $a_1, \ldots, a_n$ be the children of the root of t. Then by the construction of $\mathcal{T}$, we have $\mathsf{t}_{\geq a_i} \in \mathcal{T}_i$ and that the label of the root of t is O. Define $\mathsf{Str}(\mathsf{t}) = \mathsf{O}(\mathsf{Str}_1(\mathsf{t}_{\geq a_1}), \ldots, \mathsf{Str}_n(\mathsf{t}_{\geq a_n}))$. Using the fact that O is monotone and obeys $\mathcal{L}$-composition, and using Lemma B.1, it is easy to verify that Str is indeed an effective $\mathcal{L}$-good representation map.

(3) Since Str is effective and $\mathcal{L}$-good, by Theorem 4.2, there is a linear time f.p.t. algorithm for $\mathsf{MC}(\mathcal{L}, \mathcal{Z})$ that decides, for every $\mathcal{L}$ sentence φ, if a given structure $\mathfrak{A}$ in $\mathcal{Z}$ satisfies φ, provided that $\mathfrak{A}$ is given in the form of a tree representation of it under Str. Clearly the same algorithm is also f.p.t. for $\mathsf{MC}(\mathcal{L}, \mathcal{S})$. ◀

C Proof sketches for results concerning logical fractals

A. Proof sketch for Proposition 6.2. The following lemma is central to the proof of Proposition 6.2. The proof of the lemma uses similar ideas as used in proving Theorem 4.2; The details are presented in [29].

▶ **Lemma C.1.** *Let $\mathcal{S}$ be a class of structures that admits an $\mathcal{L}$-good tree representation $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$. Then there exists a strictly increasing computable function $\eta : \mathbb{N} \to \mathbb{N}$ such that for each $m \in \mathbb{N}$ and for each tree $\mathsf{t} \in \mathcal{T}$ of size $> \eta(m)$, there exists a proper subtree s of t in $\mathcal{T}$ such that* (i) $\mathsf{Str}(\mathsf{s}) \hookrightarrow \mathsf{Str}(\mathsf{t})$, (ii) $\mathsf{Str}(\mathsf{s}) \equiv_{m,\mathcal{L}} \mathsf{Str}(\mathsf{t})$, *and* (iii) $|\mathsf{t}| - |\mathsf{s}| \leq \eta(m)$.

Proof of Proposition 6.2. Let $\mathsf{Str} : \mathcal{T} \to \mathcal{S}$ be an $\mathcal{L}$-great representation. Then Str is $\mathcal{L}$-good, whereby by Lemma C.1, there exists a function η satisfying the properties mentioned in the lemma. For $m \in \mathbb{N}$, define $f(m) = \max\{|\mathsf{Str}(\mathsf{t})| \mid |\mathsf{t}| \leq \eta(m)\}$. Since Str is $\mathcal{L}$-great, there exists a function $\beta : \mathbb{N} \to \mathbb{N}$ satisfying the properties mentioned in the definition of $\mathcal{L}$-greatness. Then define the function $\theta_{(\mathcal{S},\mathcal{L})} : \mathbb{N}^2_+ \to \mathbb{N}_+$ as $\theta_{(\mathcal{S},\mathcal{L})}(m)(n) = f(m) + (n-1) \cdot \beta(\eta(m))$. It is easily seen that $\theta_{(\mathcal{S},\mathcal{L})}(m)$ is a scale function. Consider $\mathfrak{A} \in \mathcal{S}$ and $m \in \mathbb{N}$, and suppose that $|\mathfrak{A}| \in \langle i \rangle_g$ where g is the function $\theta_{(\mathcal{S},\mathcal{L})}(m)$ and $i > 1$. To show that for $j < i$, there exists a substructure $\mathfrak{B}$ of $\mathfrak{A}$ in $\mathcal{S}$ such that $|\mathfrak{B}| \in \langle j \rangle_g$ and $\mathfrak{B} \equiv_{m,\mathcal{L}} \mathfrak{A}$, we observe that it suffices to show the same simply for $j = i - 1$. Let $\mathsf{t} \in \mathcal{T}$ be such that $\mathsf{Str}(\mathsf{t}) = \mathfrak{A}$. By Lemma C.1, there exists a subtree s of t in $\mathcal{T}$ such that $\mathsf{Str}(\mathsf{s}) \hookrightarrow \mathsf{Str}(\mathsf{t})$, $\mathsf{Str}(\mathsf{s}) \equiv_{m,\mathcal{L}} \mathsf{Str}(\mathsf{t})$ and $|\mathsf{t}| - |\mathsf{s}| \leq \eta(m)$. Since Str is $\mathcal{L}$-great, it follows that $|\mathsf{Str}(\mathsf{t})| - |\mathsf{Str}(\mathsf{s})| \leq \beta(\eta(m))$. Whereby, either $|\mathsf{Str}(\mathsf{s})| \in \langle i-1 \rangle_g$ or $|\mathsf{Str}(\mathsf{s})| \in \langle i \rangle_g$. If the former holds, then taking $\mathfrak{B} = \mathsf{Str}(\mathsf{s})$, we are done. If the latter holds, then applying Lemma C.1 recursively to s, we eventually get

a subtree x of t in $\mathcal{T}$ such that $\mathsf{Str(x)} \hookrightarrow \mathsf{Str(t)}$, $\mathsf{Str(x)} \equiv_{m,\mathcal{L}} \mathsf{Str(t)}$ and $|\mathsf{Str(x)}| \in \langle i-1 \rangle_g$. Whereby, taking $\mathcal{B} = \mathsf{Str(x)}$, we are done. ◀

B. Proof sketch for closure properties. We finally show that the logical fractal property remains closed under the examples of operations seen earlier: the unary operations of complementation, transpose and the line-graph operation, the binary sum-like operations of disjoint union and join, and the binary product-like operations of Cartesian, tensor, strong and lexicographic products. The proofs of each of these are along the same lines as the corresponding proofs that show the closure of $\mathcal{L}$-EBSP under these operations, as given by Theorem 5.3(1). We give below the witness functions for each of these operations.

For $i \in \{1,2\}$, let $\mathcal{S}_i$ be an $\mathcal{L}$-fractal and let $\theta_{(\mathcal{S}_i,\mathcal{L})}$ be a witness to the $\mathcal{L}$-fractal property of $\mathcal{S}_i$. Let O be one of the operations mentioned above, and let $\mathcal{S}$ be the class $\{O(\mathfrak{A}) \mid \mathfrak{A} \in \mathcal{S}_1\}$ if O is unary, and the class $\{O(\mathfrak{A},\mathfrak{B}) \mid \mathfrak{A} \in \mathcal{S}_1, \mathfrak{B} \in \mathcal{S}_2\}$ if O is binary. Then $\mathcal{S}$ is an $\mathcal{L}$-fractal with witness function $\theta_{(\mathcal{S},\mathcal{L})}$ given as follows:

1. O is unary:
   a. If O is complementation or transpose, then $\theta_{(\mathcal{S},\mathcal{L})} = \theta_{(\mathcal{S}_1,\mathcal{L})}$.
   b. If O is the line-graph operation and $\mathcal{L} = \mathrm{FO}$, then $\theta_{(\mathcal{S},\mathcal{L})}(m)(n) = (\theta_{(\mathcal{S}_1,\mathcal{L})}(m)(n))^2$ for all $m, n \in \mathbb{N}_+$.
2. O is binary and sum-like: $\theta_{(\mathcal{S},\mathcal{L})}(m)(n) = \theta_{(\mathcal{S}_1,\mathcal{L})}(m)(1) + \theta_{(\mathcal{S}_2,\mathcal{L})}(m)(n)$ for all $m, n \in \mathbb{N}_+$.
3. O is binary and product-like, and $\mathcal{L} = \mathrm{FO}$: We define $\theta_{(\mathcal{S},\mathcal{L})}(m)(n)$ inductively as follows for all $m, n \in \mathbb{N}_+$:
   - $\theta_{(\mathcal{S},\mathcal{L})}(m)(1) = \theta_{(\mathcal{S}_1,\mathcal{L})}(m)(1) \cdot \theta_{(\mathcal{S}_2,\mathcal{L})}(m)(1)$
   - Let $n > 1$. For $i \in \{1,2\}$, let $k_i = \min(\{j \mid \theta_{(\mathcal{S}_i,\mathcal{L})}(m)(j) \geq \theta_{(\mathcal{S},\mathcal{L})}(m)(n)\})$. Then $\theta_{(\mathcal{S},\mathcal{L})}(m)(n+1) = \theta_{(\mathcal{S}_1,\mathcal{L})}(m)(k_1+1) \cdot \theta_{(\mathcal{S}_2,\mathcal{L})}(m)(k_2+1)$.

For unary and sum-like binary operations, it is easy to verify that $\theta_{(\mathcal{S},\mathcal{L})}$ indeed witnesses the $\mathcal{L}$-fractal property of $\mathcal{S}$. For product-like binary operations, here is the explanation. For $m \in \mathbb{N}_+$, let f be the function $\theta_{(\mathcal{S},\mathcal{L})}(m)$. Consider $\mathfrak{C} \in \mathcal{S}$ such that $|\mathfrak{C}| \in \langle i \rangle_f$. We assume $i > 2$; the $i = 2$ case can be done similarly. Then $\mathfrak{C} = O(\mathfrak{A},\mathfrak{B})$ for $\mathfrak{A} \in \mathcal{S}_1$ and $\mathfrak{B} \in \mathcal{S}_2$. By construction of $\theta_{(\mathcal{S},\mathcal{L})}$, we have $|\mathfrak{C}| = |\mathfrak{A}| \cdot |\mathfrak{B}| \geq f(i-1) = \theta_{(\mathcal{S}_1,\mathcal{L})}(m)(k_1+1) \cdot \theta_{(\mathcal{S}_2,\mathcal{L})}(m)(k_2+1)$, where $k_i = \min(\{j \mid \theta_{(\mathcal{S}_i,\mathcal{L})}(m)(j) \geq f(i-2)\})$ for $i \in \{1,2\}$. We have two cases here:

1. $|\mathfrak{A}| \geq \theta_{(\mathcal{S}_1,\mathcal{L})}(m)(k_1+1)$ and $|\mathfrak{B}| \geq \theta_{(\mathcal{S}_2,\mathcal{L})}(m)(k_2+1)$: Then since $\mathcal{S}_i$ is an $\mathcal{L}$-fractal with witness $\theta_{(\mathcal{S}_i,\mathcal{L})}$ for $i \in \{1,2\}$, there exists a substructure $\mathfrak{A}'$ of $\mathfrak{A}$ in $\mathcal{S}_1$, resp. substructure $\mathfrak{B}'$ of $\mathfrak{B}$ in $\mathcal{S}_2$ such that $\mathfrak{A}' \equiv_{m,\mathcal{L}} \mathfrak{A}$ and $\theta_{(\mathcal{S}_1,\mathcal{L})}(m)(k_1) < |\mathfrak{A}'| \leq \theta_{(\mathcal{S}_1,\mathcal{L})}(m)(k_1+1)$, resp. $\mathfrak{B}' \equiv_{m,\mathcal{L}} \mathfrak{B}$ and $\theta_{(\mathcal{S}_2,\mathcal{L})}(m)(k_2) < |\mathfrak{B}'| \leq \theta_{(\mathcal{S}_2,\mathcal{L})}(m)(k_2+1)$. Let $\mathfrak{C}' = O(\mathfrak{A}',\mathfrak{B}')$.
2. Assume w.l.o.g. that $|\mathfrak{A}| \geq \theta_{(\mathcal{S}_1,\mathcal{L})}(m)(k_1+1)$ and $|\mathfrak{B}| < \theta_{(\mathcal{S}_2,\mathcal{L})}(m)(k_2+1)$. Then consider the structure $\mathfrak{A}'$ as described above, and let $\mathfrak{C}' = O(\mathfrak{A}',\mathfrak{B})$.

In either case, we observe that $\mathfrak{C}'$ is a substructure of $\mathfrak{C}$ in $\mathcal{S}$ such that $\mathfrak{C}' \equiv_{m,\mathcal{L}} \mathfrak{C}$ and $|\mathfrak{C}'| \in \langle i-1 \rangle_f$. Whereby, $\mathcal{S}$ is an $\mathcal{L}$-fractal, and $\theta_{(\mathcal{S},\mathcal{L})}$ witnesses the $\mathcal{L}$-fractal property of $\mathcal{S}$.

$\aleph_1$ and the Modal μ-Calculus*

Maria João Gouveia[†1] **and Luigi Santocanale**[2]

1 **CEMAT-CIÊNCIAS (UID/Multi/04621/2013), Faculdade de Ciências, Universidade de Lisboa, Lisbon, Portugal**
mjgouveia@fc.ul.pt

2 **Laboratoire d'Informatique Fondamentale de Marseille, UMR 7279, CNRS AMU, Marseille, France**
luigi.santocanale@lif.univ-mrs.fr

Abstract

For a regular cardinal κ, a formula of the modal μ-calculus is κ-continuous in a variable x if, on every model, its interpretation as a unary function of x is monotone and preserves unions of κ-directed sets. We define the fragment $\mathcal{C}_{\aleph_1}(x)$ of the modal μ-calculus and prove that all the formulas in this fragment are $\aleph_1$-continuous. For each formula $\phi(x)$ of the modal μ-calculus, we construct a formula $\psi(x) \in \mathcal{C}_{\aleph_1}(x)$ such that $\phi(x)$ is κ-continuous, for some κ, if and only if $\phi(x)$ is equivalent to $\psi(x)$. Consequently, we prove that (i) the problem whether a formula is κ-continuous for some κ is decidable, (ii) up to equivalence, there are only two fragments determined by continuity at some regular cardinal: the fragment $\mathcal{C}_{\aleph_0}(x)$ studied by Fontaine and the fragment $\mathcal{C}_{\aleph_1}(x)$. We apply our considerations to the problem of characterizing closure ordinals of formulas of the modal μ-calculus. An ordinal α is the closure ordinal of a formula $\phi(x)$ if its interpretation on every model converges to its least fixed-point in at most α steps and if there is a model where the convergence occurs exactly in α steps. We prove that ω_1, the least uncountable ordinal, is such a closure ordinal. Moreover we prove that closure ordinals are closed under ordinal sum. Thus, any formal expression built from $0, 1, \omega, \omega_1$ by using the binary operator symbol $+$ gives rise to a closure ordinal.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Modal μ-calculus, regular cardinal, continuous function, $\aleph_1$, ω_1, closure ordinal, ordinal sum

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.38

1 Introduction

The propositional modal μ-calculus [17, 20] is a well established logic in theoretical computer science, mainly due to its convenient properties for the verification of computational systems. It includes as fragments many other computational logics, PDL, CTL, CTL*, its expressive power is therefore highly appreciated. Also, being capable to express all the bisimulation invariant properties of transition systems that are definable in monadic second order logic, the modal μ-calculus can itself be considered as a robust fragment of an already very expressive logic [14]. Despite its strong expressive power, this logic is still considered as a tractable one: its model checking problem, even if in the class UP $\cap$ co-UP [15], becomes polynomial as soon as some critical parameters are fixed or restricted classes of models are considered [22, 3, 5]. The widespread interest for this logic has triggered further researches

* Full version of this paper is available from Hal: https://hal.archives-ouvertes.fr/hal-01503091.
† Partially supported by FCT under grant SFRH/BSAB/128039/2016.

that spread beyond the realm of verification: these concern the expressive power [7, 4], axiomatic bases [30], algebraic and order theoretic approaches [26], deductive systems [21, 27] and the semantics of functional programs [11].

The present paper lies at the intersection of two lines of research on the modal μ-calculus, on continuity [10] and on closure ordinals [9, 2]. Continuity of monotone functions is a fundamental phenomenon in modal logic, on which well known uniform completeness theorems rely [24, 12, 16]. Fontaine [10] characterized the formulas of the modal μ-calculus that give rise to continuous functions on Kripke models. It is well known, for example in categorical approaches to model theory [1], that the notion of continuity of monotone functions (and of functors) can be generalized to κ-continuity, where the parameter κ is an infinite regular cardinal. In the work [25] one of the authors proved that $\aleph_1$-continuous functors are closed under their greatest fixed-points. Guided by this result, we present in this paper a natural syntactic fragment $\mathcal{C}_{\aleph_1}(x)$ of the modal μ-calculus whose formulas are $\aleph_1$-continuous—that is, they give rise to $\aleph_1$-continuous monotone unary functions of the variable x on arbitrary models. A first result that we present here is that *the fragment $\mathcal{C}_{\aleph_1}(x)$ is decidable*: for each $\phi(x) \in \mathsf{L}_\mu$, we construct a formula $\psi(x) \in \mathcal{C}_{\aleph_1}(x)$ such that $\phi(x)$ is $\aleph_1$-continuous on every model if and only if $\phi(x)$ and $\psi(x)$ are semantically equivalent formulas. We borrow some techniques from [10], yet the construction of the formula $\psi(x)$ relies on a new notion of normal form for formulas of the modal μ-calculus. A closer inspection of our proof uncovers a stronger fact: the formulas $\phi(x)$ and $\psi(x)$ are equivalent if and only if, for some regular cardinal κ, $\phi(x)$ is κ-continuous on every model. The stronger statement implies that we cannot find a fragment $\mathcal{C}_\kappa(x)$ of κ-continuous formulas for some cardinal κ strictly larger than $\aleph_1$; any such hypothetical fragment collapses, semantically, to the fragment $\mathcal{C}_{\aleph_1}(x)$.

Our interest in $\aleph_1$-continuity was wakened once more when researchers started investigating closure ordinals of formulas of the modal μ-calculus [9, 2]. Indeed, we consider closure ordinals as a wide field where the notion of κ-continuity can be exemplified and applied; the two notions, κ-continuity and closure ordinals, are naturally intertwined. An ordinal α is the closure ordinal of a formula $\phi(x)$ if (the interpretation of) this formula (as a monotone unary function of the variable x) converges to its least fixed-point $\mu_x.\phi(x)$ in at most α steps in every model and, moreover, there exists at least one model in which the formula converges exactly in α steps. Not every formula has a closure ordinal. For example, the simple formula $[\,]x$ has no closure ordinal; more can be said, this formula is not κ-continuous for any κ. As a matter of fact, if a formula $\phi(x)$ is κ-continuous (that is, if its interpretation on every model is κ-continuous), then it has a closure ordinal $\mathsf{cl}(\phi(x)) \leqslant \kappa$—here we use the fact that, using the axiom of choice, a cardinal can be identified with a particular ordinal, for instance $\aleph_0 = \omega$ and $\aleph_1 = \omega_1$. Our results on $\aleph_1$-continuity shows that all the formulas in $\mathcal{C}_{\aleph_1}(x)$ have a closure ordinal bounded by ω_1. For closure ordinals, our results are threefold. Firstly we prove that *the least uncountable ordinal ω_1 belongs to the set* $\mathrm{Ord}(\mathsf{L}_\mu)$ of all closure ordinals of formulas of the propositional modal μ-calculus. Secondly, we prove that $\mathrm{Ord}(\mathsf{L}_\mu)$ *is closed under ordinal sum.* It readily follows that any formal expression built from $0, 1, \omega, \omega_1$ by using the binary operator symbol $+$ gives rise to an ordinal in $\mathrm{Ord}(\mathsf{L}_\mu)$. Let us recall that Czarnecki [9] proved that all the ordinals $\alpha < \omega^2$ belong to $\mathrm{Ord}(\mathsf{L}_\mu)$. Our results generalize Czarnecki's construction of closure ordinals and give it a rational reconstruction—every ordinal strictly smaller than ω^2 can be generated by $0, 1$ and ω by repeatedly using the sum operation. Finally, the fact that there are no relevant fragments of the modal μ-calculus determined by continuity at some regular cardinal other than $\aleph_0$ and $\aleph_1$ implies that *the methodology* (adding regular cardinals to $\mathrm{Ord}(\mathsf{L}_\mu)$ and closing them under ordinal sum) *used until now to construct new closure ordinals* for the modal μ-calculus *cannot be further exploited.*

Let us add some final considerations. In our view, the discovery of the fragment $\mathcal{C}_{\aleph_1}(x)$ opens an unsuspected new dimension (thus new tools, new ideas, new perspectives, etc.) in the theory of the modal μ-calculus and of fixed-point logics. Consider for example the modal μ-calculus on deterministic models, where states have at most one successor; we immediately obtain that every formula is $\aleph_1$-continuous on these models. Whether this and other observations can be exploited (towards understanding alternation hierarchies or reasoning using axiomatic bases, for example) is part of future researches. Yet we believe that the scopes of this work and of the problems studied here go well beyond the pure theory of the modal μ-calculus. Our interest in closure ordinals stems from a proof-theoretic work on induction and coinduction [11, 25]. There we banned ordinal notations from the syntax, as we considered the theory of ordinals too strong for our constructive goals. Yet our judgement might have gone too far, since the theory needed to deal with ordinals is not that strong; for example, many statements on ordinals do not need the axiom of choice. This makes reasonable to devise syntaxes based on ordinals. With respect to these problems, related to the semantics of programming languages, the closure ordinal problem becomes an optimal playground where to develop and test intuitions.

The paper is structured as follows. In Section 2 we introduce the notion of κ-continuity and illustrate its interactions with fixed-points. In Section 3 we present the modal μ-calculus and some tools that shall be needed in the following sections. Section 4 presents our results on the fragment $\mathcal{C}_{\aleph_1}(x)$. In Section 5 we argue that the least uncountable ordinal is a closure ordinal for the modal μ-calculus and that $\mathrm{Ord}(\mathsf{L}_\mu)$ is closed under ordinal sum.

Proof of all the statements can be found in the preprint [13].

2 κ-continuous mappings and their extremal fixed-points

In this section we consider κ-continuity of mappings between powerset Boolean algebras, where the parameter κ is an infinite regular cardinal. If $\kappa = \aleph_0$, then κ-continuity coincides with the usual notion of continuity as known, for example, from [10]. The interested reader might find further informations in the monograph [1]. In the second part of this section we recall how κ-continuity interacts with least and greatest fixed-points.

In the following κ is a fixed infinite regular cardinal, $P(A)$ and $P(B)$ are the powerset Boolean algebras, for some sets A and B, and $f : P(A) \to P(B)$ is a monotone mapping.

▶ **Definition 1.** A subset $\mathcal{I} \subseteq P(A)$ is a *κ-directed set* if every collection $\mathcal{J} \subseteq \mathcal{I}$ with $\operatorname{card} \mathcal{J} < \kappa$ has an upper bound in $\mathcal{I}$. A mapping $f : P(A) \to P(B)$ is *κ-continuous* if $f(\bigcup \mathcal{I}) = \bigcup f(\mathcal{I})$, whenever $\mathcal{I}$ is a κ-directed set.

▶ Remark. If κ' is a regular cardinal and $\kappa < \kappa'$, then a κ'-directed set is also a κ-directed set. Whence, if f is κ-continuous, then it also preserves unions of κ'-directed sets, thus it is also κ'-continuous.

We shall say that a subset X of A is *κ-small* if $\operatorname{card} X < \kappa$. For example, a set X is $\aleph_0$-small if and only if it is finite, and it is $\aleph_1$-small if and only if it is countable.

▶ **Proposition 2.** *For each $X \subseteq A$, X is κ-small if and only if, for every κ-directed set $\mathcal{I}$, $X \subseteq \bigcup \mathcal{I}$ implies $X \subseteq I$, for some $I \in \mathcal{I}$.*

▶ **Proposition 3.** *A monotone mapping $f : P(A) \to P(B)$ is κ-continuous if and only if, for every $X \in P(A)$,*

$$f(X) = \bigcup \{ f(X') \mid X' \subseteq X, X' \text{ is } \kappa\text{-small} \}.$$

Proof. Let $f : P(A) \to P(B)$ be a monotone mapping and suppose that f is κ-continuous. In $P(A)$ every element X is the union of the set $\mathcal{I}_\kappa(X) := \{ X' \mid X' \subseteq X, X' \text{ is } \kappa\text{-small} \}$ which is a κ-directed set: just observe that if $\mathcal{J} \subseteq P(A)$ is a κ-small collection of κ-small subsets of A, then $\bigcup \mathcal{J}$ is κ-small, since the cardinal κ is regular. Then $f(X) = f(\bigcup \mathcal{I}_\kappa(X)) = \bigcup f(\mathcal{I}_\kappa(X))$.

Conversely suppose that $f : P(A) \to P(B)$ is a monotone mapping such that $f(X) = \bigcup f(\mathcal{I}_\kappa(X))$ for every $X \in P(A)$. Let $\mathcal{I}$ be a κ-ideal and let X' be a κ-small set contained in $\bigcup \mathcal{I}$. By Proposition 2 there exists $I \in \mathcal{I}$ such that $X' \subseteq I$. But then $X' \in \mathcal{I}$ since $\mathcal{I}$ is a downward closed set. Thus $\mathcal{I}_\kappa(\bigcup \mathcal{I}) \subseteq \mathcal{I}$ and consequently $f(\bigcup \mathcal{I}) = \bigcup f(\mathcal{I}_\kappa(\bigcup \mathcal{I})) \subseteq \bigcup f(\mathcal{I})$. Since $\bigcup f(\mathcal{I}) \subseteq f(\bigcup \mathcal{I})$ we obtain $f(\bigcup \mathcal{I}) = \bigcup f(\mathcal{I})$. ◀

2.1 Fixed-points of κ-continuous mappings

The Knaster-Tarski theorem [28] states that if $f : P(A) \to P(A)$ is monotone, then the set $\bigcap \{ X \subseteq A \mid f(X) \subseteq X \}$ is the least fixed-point of f. On the other hand, Kleene's fixed-point theorem states that least fixed-point of an $\aleph_0$-continuous mapping f is constructible by iterating ω_0-times f starting from the empty set, namely it is equal to $\bigcup_{n \geq 0} f^n(\emptyset)$. Generalizations of Kleene's theorem, constructing the least fixed-point of a monotone f by ordinal approximations, appeared later, see for example [8], [19]. The following Proposition 5 generalizes Kleene's theorem to κ-continuous mappings. To state it, we firstly introduce the notions of approximant and convergence.

▶ **Definition 4.** If $f : P(A) \to P(A)$ is a monotone function, then the *approximants* $f^\alpha(\emptyset)$, α an ordinal, are inductively defined as follows:

$$f^{\alpha+1}(\emptyset) := f(f^\alpha(\emptyset)), \qquad f^\alpha(\emptyset) := \bigcup_{\beta<\alpha} f^\beta(\emptyset) \quad \text{when } \alpha \text{ is a limit ordinal.}$$

We say that f *converges to its least fixed-point in at most* α *steps* if $f^\alpha(\emptyset)$ is a fixed-point (necessarily the least one) of f. We say that f *converges to its least fixed-point in exactly* α *steps* if $f^\alpha(\emptyset)$ is a fixed-point of f and $f^\beta(\emptyset) \subsetneq f^{\beta+1}(\emptyset)$, for each ordinal $\beta < \alpha$.

Let us recall that in set theory a cardinal κ is identified with the least ordinal of cardinality equal to κ. We exploit this, notationally, in the next proposition.

▶ **Proposition 5.** *If $f : P(A) \to P(A)$ is a κ-continuous monotone function, then it converges to its least fixed-point in at most κ steps.*

Proof. Let us argue that $f^\kappa(\emptyset)$ is a fixed-point of f:

$$f(\, f^\kappa(\emptyset)\,) = f(\, \bigcup_{\alpha<\kappa} f^\alpha(\emptyset)\,) = \bigcup_{\alpha<\kappa} f(f^\alpha(\emptyset)) \subseteq \bigcup_{\alpha<\kappa} f^\alpha(\emptyset) = f^\kappa(\emptyset)$$

since the regularity of κ implies that $\{ f^\alpha(\emptyset) \mid \alpha < \kappa \}$ is a κ-directed set. ◀

Propositions 6 and 7 are specific instances of a result stated for categories [25]. In order to clarify their statements, we first observe that if $f : P(B) \times P(A) \to P(B)$ is a monotone mapping, then the unary mapping $f(-, X) : P(B) \to P(B)$, $Z \mapsto f(Z, X)$, is also monotone. Hence we may consider the mapping $P(A) \to P(A)$ that sends X to the least (resp. greatest) fixed-point of $f(-, X)$; by using the standard μ-calculus notation, we denote it by $\mu_z.f(z, -)$ (resp. $\nu_z.f(z, -)$). We also recall that f is κ-continuous w.r.t. the coordinate-wise order on $P(B) \times P(A)$ if and only if it is κ-continuous in every variable.

▶ **Proposition 6.** *Let $f : P(B) \times P(A) \to P(B)$ be a κ-continuous monotone mapping. If $\kappa > \aleph_0$ then $\nu_z.f(z, \_) : P(A) \to P(B)$ is also κ-continuous.*

Proof. Let us write $g(x) := \nu_z.f(z,x)$. We shall show that, for every $b \in B$ and for $X \in P(A)$, if $b \in g(X)$, then $b \in g(X')$ for some κ-small X' contained in X. Having shown this, it follows by Proposition 3 that g is continuous. Note that the condition $b \in g(X)$ holds when there exists $Z \subseteq B$ such that $b \in Z$ and $Z \subseteq f(Z,X)$. Aiming to find such a set Z we recursively obtain a family $(X_n)_{n\geq 1}$ of κ-small subsets of X and a family $(Z_n)_{n\geq 0}$ of κ-small subsets of Z satisfying $Z_n \subseteq f(Z_{n+1}, X_{n+1})$.

For $n = 0$ we take $Z_0 := \{\, b \,\}$ which is a κ-small subset of $f(Z,X)$. Now suppose we have already constructed Z_n which is κ-small and satisfies $Z_n \subseteq f(Z,X)$. Let us consider

$$\mathcal{I} := \{\, f(Z',X') \mid X' \subseteq X, Z' \subseteq Z \text{ and } X', Z' \kappa\text{-small} \,\}\,.$$

Since $Z_n \subseteq f(Z,X) = \bigcup \mathcal{I}$ and $\mathcal{I}$ is a κ-directed set, by Proposition 2 there exist Z_{n+1}, X_{n+1} κ-small such that $Z_n \subseteq f(Z_{n+1}, X_{n+1})$. Moreover, $Z_{n+1} \subseteq Z \subseteq f(Z,X)$.

Let now $X_\omega := \bigcup_{n\geq 1} X_n$ and $Z_\omega := \bigcup_{n\geq 0} Z_n$. Notice that Z_ω and X_ω are κ-small, since we assume that $\kappa > \aleph_0$. We have therefore

$$Z_\omega = \bigcup_{n\geq 0} Z_n \subseteq \bigcup_{n\geq 1} f(Z_n, X_n) \subseteq f(\bigcup_{n\geq 1} Z_n, \bigcup_{n\geq 1} X_n) \subseteq f(Z_\omega, X_\omega)\,.$$

Whence $b \in Z_\omega \subseteq \nu_z.f(z, X_\omega)$, with $X_\omega \subseteq X$ and X_ω κ-small, proving that $\nu_z.f(z,-)$ is κ-continuous. ◀

▶ **Proposition 7.** *Suppose that $\kappa \geq \aleph_0$ and let $f : P(B) \times P(A) \to P(B)$ be a κ-continuous monotone mapping. Then $\mu_z.f(z,\_) : P(A) \to P(B)$ is also κ-continuous.*

3 The propositional modal μ-calculus

In this section we present the propositional modal μ-calculus and some known results on this logic that we shall need later.

Henceforward *Act* is a fixed finite set of actions and *Prop* is a countable set of propositional variables. The set L_μ of formulas of the propositional modal μ-calculus over *Act* is generated by the following grammar:

$$\phi := y \mid \neg y \mid \top \mid \phi \wedge \phi \mid \bot \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a]\phi \mid \mu_z.\phi \mid \nu_z.\phi\,,$$

where $a \in Act$, $y \in Prop$, and $z \in Prop$ is a positive variable in the formula ϕ, i.e. no occurrence of z is under the scope of a negation. We assume that *Prop* contains variables $x, x_1, \ldots, x_n, \ldots$ that are never under the scope of a negation nor bound in a formula ϕ. In general, we shall use $y, y_1, \ldots y_n, \ldots$ for variables that are free in formulas, and $z, z_1, \ldots, z_n, \ldots$ for variables that are bound in formulas.

An *Act-model* (hereinafter referred to as model) is a triple $\mathcal{M} = \langle |\mathcal{M}|, \{\, R_a \mid a \in Act \,\}, v \rangle$ where $|\mathcal{M}|$ is a set, $R_a \subseteq |\mathcal{M}| \times |\mathcal{M}|$ for each $a \in Act$, and $v : Prop \to P(|\mathcal{M}|)$ is an interpretation of the propositional variables as subsets of $|\mathcal{M}|$. Given a model $\mathcal{M}$, the semantics $[\![\psi]\!]_{\mathcal{M}}$ of formulas $\psi \in \mathsf{L}_\mu$ as subsets of $|\mathcal{M}|$ is recursively defined using the standard clauses from polymodal logic **K**. For example, we have

$$[\![\langle a \rangle \psi]\!]_{\mathcal{M}} = \{\, s \in |\mathcal{M}| \mid \exists s' \text{ s.t. } sR_as' \text{ and } \in [\![\psi]\!]_{\mathcal{M}} \,\}\,,$$
$$[\![[a]\psi]\!]_{\mathcal{M}} = \{\, s \in |\mathcal{M}| \mid \forall s' sR_as' \text{ implies } s' \in [\![\psi]\!]_{\mathcal{M}} \,\}\,.$$

Here we only define the semantics of the least and greatest fixed-point constructors μ and ν. To this goal, given a subset $Z \subseteq |\mathcal{M}|$, we define $\mathcal{M}[Z/z]$ to be the model that differs

from $\mathcal{M}$ only on the value Z that its valuation takes on z. The clauses for the fixed-point constructors are the following:

$$[\![\mu_z.\psi]\!]_{\mathcal{M}} := \bigcap \{\, Z \subseteq |\mathcal{M}| \mid [\![\psi]\!]_{\mathcal{M}[Z/z]} \subseteq Z \,\},$$
$$[\![\nu_z.\psi]\!]_{\mathcal{M}} := \bigcup \{\, Z \subseteq |\mathcal{M}| \mid Z \subseteq [\![\psi]\!]_{\mathcal{M}[Z/z]} \,\}.$$

A formula $\phi \in \mathsf{L}_\mu$ and a variable $x \in \mathit{Prop}$ determine on every model $\mathcal{M}$ the correspondence $\phi^x_{\mathcal{M}} : P(|\mathcal{M}|) \to P(|\mathcal{M}|)$, that sends each $S \subseteq |\mathcal{M}|$ to $[\![\phi]\!]_{\mathcal{M}[S/x]} \subseteq |\mathcal{M}|$—in the following we shall write $\phi_{\mathcal{M}}$ for $\phi^x_{\mathcal{M}}$, when x is understood. Due to the syntactic restriction on the variable z in the productions of $\mu_z.\phi$ and $\nu_z.\phi$, the function $\phi^z_{\mathcal{M}}$ is monotone. By Tarski's theorem [28], the above clauses state that $[\![\mu_z.\phi]\!]_{\mathcal{M}}$ and $[\![\nu_z.\phi]\!]_{\mathcal{M}}$ are, respectively, the least and the greatest fixed-point of $\phi^z_{\mathcal{M}}$. As usual, we write $\mathcal{M}, s \Vdash \psi$ to mean that $s \in [\![\psi]\!]_{\mathcal{M}}$.

The closure of a formula

For $\phi \in \mathsf{L}_\mu$, we denote by $\mathit{Sub}(\phi)$ the set of subformulas of ϕ. For $\psi \in \mathit{Sub}(\phi)$, the *standard context* of ψ in ϕ is the (composed) substitution

$$\sigma^\phi_\psi := [Q^n_{z_n}.\psi_n/z_n] \cdot \cdots \cdot [Q^1_{z_1}.\psi_1/z_1]$$

uniquely determined by the following conditions:

1. $\{ z_1, \ldots, z_n \}$ is the set of variables bound in ϕ and free in ψ,
2. for each $i = 1, \ldots, n$, $Q^i_{z_i}.\psi_i$ is the unique subformula of ϕ such that $Q^i \in \{ \mu, \nu \}$,
3. $Q^j_{z_j}.\psi_j$ is a subformula of ψ_i, for $i < j$.

For $\phi \in \mathsf{L}_\mu$, the *closure* of ϕ, see [17], is the set $\mathrm{CL}(\phi)$ defined as follows:

$$\mathrm{CL}(\phi) := \{\, \psi \cdot \sigma^\phi_\psi \mid \psi \in \mathit{Sub}(\phi) \,\}.$$

Recall from [17] that $\mathrm{CL}(\phi)$ is the least subset of L_μ such that

- $\phi \in \mathrm{CL}(\phi)$,
- if $\psi_1 @ \psi_2 \in \mathrm{CL}(\phi)$, then $\psi_1, \psi_2 \in \mathrm{CL}(\phi)$, with $@ \in \{ \wedge, \vee \}$,
- if $\langle a \rangle \psi \in \mathrm{CL}(\phi)$ or $[a]\psi \in \mathrm{CL}(\phi)$, then $\psi \in \mathrm{CL}(\phi)$,
- if $Q_z.\psi \in \mathrm{CL}(\phi)$, then $\psi[\mu_z.\psi/z] \in \mathrm{CL}(\phi)$, with $Q \in \{ \mu, \nu \}$.

By definition, $\mathrm{CL}(\phi)$ is finite. The characterization of $\mathrm{CL}(\phi)$ as the least subset satisfying the above conditions yields the following observation: if $\psi \in \mathrm{CL}(\phi)$, then $\mathrm{CL}(\psi) \subseteq \mathrm{CL}(\phi)$.

Game semantics

Given $\phi \in \mathsf{L}_\mu$ and a model $\mathcal{M} = \langle |\mathcal{M}|, \{ R_a \mid a \in \mathit{Act} \}, v \rangle$, the game $\mathcal{G}(\mathcal{M}, \phi)$ has $|\mathcal{M}| \times \mathrm{CL}(\phi)$ as its set of positions. Moves are as in the table below:

Adam's moves	Eva's moves
$(s, \psi_1 \wedge \psi_2) \to (s, \psi_i), \quad i = 1, 2$	$(s, \psi_1 \vee \psi_2) \to (s, \psi_i), \quad i = 1, 2,$
$(s, [a]\psi) \to (s', \psi), \quad sR_as'$	$(s, \langle a \rangle \psi) \to (s', \psi), \quad sR_as',$
$(s, \nu_z.\psi) \to (s, \psi[\nu_z.\psi/z])$	$(s, \mu_z.\psi) \to (s, \psi[\mu_z.\psi/z]).$

From a position of the form $(s, \top)$ Adam loses, and from a position of the form $(s, \bot)$ Eva loses. Also, from a position of the form (s, p) with p a propositional variable, Eva wins if and

only if $s \in v(p)$; from a position of the form $(s, \neg p)$ with p a propositional variable, Eva wins if and only if $s \notin v(p)$. The definition of the game is completed by wins on infinite plays. To this goal we choose a rank function $\rho : \mathrm{CL}(\phi) \to \mathbb{N}$ such that, when ψ_1 is a subformula of ψ_2, then $\rho(\psi_1 \cdot \sigma^{\phi}_{\psi_1}) \leq \rho(\psi_2 \cdot \sigma^{\phi}_{\psi_2})$, and such that $\rho(\mu_z.\psi)$ is odd and $\rho(\nu_z.\psi)$ is even. An infinite play $\{ (s_n, \psi_n) \mid n \geq 0 \}$ is a win for Eva if and only if $\max\{ n \geq 0 \mid \{ i \mid \rho^{-1}(\psi_i) \text{ is infinite} \} \}$ is even. Let us recall the following fundamental result (see for example [6, Theorem 6]):

▶ **Proposition 8.** *For each model $\mathcal{M}$ and each formula $\phi \in \mathsf{L}_\mu$, $\mathcal{M}, s \Vdash \phi$ if and only if Eva has a winning strategy from position (s, ϕ) in the game $\mathcal{G}(\mathcal{M}, \phi)$.*

Bisimulations

Let $P \subseteq Prop$ be a subset of variables and let $B \subseteq Act$ be a subset of actions. Let $\mathcal{M}$ and $\mathcal{M}'$ be two models. A (P, B)-bisimulation is a relation $\mathcal{B} \subseteq |\mathcal{M}| \times |\mathcal{M}'|$ such that, for all $(x, x') \in \mathcal{B}$, we have

- $x \in v(p)$ if and only if $x' \in v'(p)$, for all $p \in P$,
- for each $b \in B$,
  - xR_by implies $x'R_by'$ for some y' such that $(y, y') \in \mathcal{B}$,
  - $x'R_by'$ implies xR_by for some y such that $(y, y') \in \mathcal{B}$.

A pointed model is a pair $\langle \mathcal{M}, s \rangle$ with $\mathcal{M} = \langle |\mathcal{M}|, \{ R_a \mid a \in Act \}, v \rangle$ a model and $s \in |\mathcal{M}|$. We say that two pointed models $\langle \mathcal{M}, s \rangle$ and $\langle \mathcal{M}', s' \rangle$ are (P, B)-bisimilar if there exists a (P, B)-bisimulation $\mathcal{B} \subseteq |\mathcal{M}| \times |\mathcal{M}|'$ with $(s, s') \in \mathcal{B}$; we say that they are bisimilar if they are $(Prop, Act)$-bisimilar.

Let us denote by $\mathsf{L}_\mu[P, B]$ the set of formulas whose free variables are in P and whose modalities are only indexed by actions in B. The following statement is a straightforward refinement of [6, Theorem 10].

▶ **Proposition 9.** *If $\langle \mathcal{M}, s \rangle$ and $\langle \mathcal{M}', s' \rangle$ are (P, B)-bisimilar, then $\mathcal{M}, s \Vdash \phi$ if and only if $\mathcal{M}', s' \Vdash \phi$, for each $\phi \in \mathsf{L}_\mu[P, B]$.*

Submodels

If $\mathcal{M} = \langle |\mathcal{M}|, \{ R_a^{\mathcal{M}} \mid a \in Act \}, v \rangle$ is a model, then a subset S of $|\mathcal{M}|$ determines the model $\mathcal{M}_{\restriction S} := \langle S, \{ R_a \cap S \times S \mid a \in Act \}, v' \rangle$ where $v'(y) = v(y) \cap S$. We call the *submodel of $\mathcal{M}$ induced* by S. A subset S of $|\mathcal{M}|$ is *closed* if $s \in S$ and sR_as' imply $s' \in S$, for every $a \in Act$.

▶ **Proposition 10.** *For each formula $\phi \in \mathsf{L}_\mu$, there exists a formula $\mathtt{tr}(\phi) \in \mathsf{L}_\mu$, containing a new propositional variable p, with the following property: for each model $\mathcal{M}$, each subset $S \subseteq |\mathcal{M}|$, and each $s \in |\mathcal{M}|$,*

$$\mathcal{M}[S/p], s \models \mathtt{tr}(\phi) \quad \textit{iff} \quad s \in S \text{ and } \mathcal{M}_{\restriction S}, s \models \phi\,.$$

Moreover, for each ordinal α, $\mathtt{tr}(\phi)^{\alpha}_{\mathcal{M}[S/p]}(\emptyset) = \phi^{\alpha}_{\mathcal{M}_{\restriction S}}(\emptyset)$.

▶ Remark. In the statement of the previous proposition, the formula $\mathtt{tr}(\phi)$ is, in general, defined by induction. Yet, if S is a closed subset of $\mathcal{M}$, then we can simply let $\mathtt{tr}(\phi) := p \wedge \phi$.

4 $\aleph_1$-continuous fragment of the modal μ-calculus

We introduce in this section the fragment $\mathcal{C}_{\aleph_1}(x)$ of the modal μ-calculus whose formulas, when interpreted as monotone functions of the variable x, are all $\aleph_1$-continuous. We show

how to construct a formula $\phi' \in \mathcal{C}_{\aleph_1}(x)$ from a given formula ϕ such that ϕ is κ-continuous, for some κ, if and only if ϕ and ϕ' are equivalent formulas. We argue therefore that the problem whether a formula is κ-continuous for some κ is decidable and, moreover, that there are no interesting notions of κ-continuity, for the modal μ-calculus, besides those for the cardinals $\aleph_0$ and $\aleph_1$.

A formula $\phi \in \mathsf{L}_\mu$ *is κ-continuous in x* if $\phi_{\mathcal{M}}$ is κ-continuous, for each model $\mathcal{M}$. If $X \subseteq \mathit{Prop}$, then we say that ϕ *is κ-continuous in X* if ϕ is κ-continuous in x, for each $x \in X$.

Define $\mathcal{C}_{\aleph_1}(X)$ to be the set of formulas of the modal μ-calculus that can be generated by the following grammar:

$$\phi := x \mid \psi \mid \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid \mu_z.\chi \mid \nu_z.\chi\,,$$

where $x \in X$, $\psi \in \mathsf{L}_\mu$ is a μ-calculus formula not containing any variable $x \in X$, and $\chi \in \mathcal{C}_{\aleph_1}(X \cup \{ z \})$. If we omit the last production from the above grammar, we obtain a grammar for the continuous fragment of the modal μ-calculus, see [10], which we denote here by $\mathcal{C}_{\aleph_0}(X)$. For $i = 0, 1$, we shall write $\mathcal{C}_{\aleph_i}(x)$ for $\mathcal{C}_{\aleph_i}(\{ x \})$. The main achievement of [10] is that a formula $\phi \in \mathsf{L}_\mu$ is $\aleph_0$-continuous in x if and only if it is equivalent to a formula in $\mathcal{C}_{\aleph_0}(x)$.

Observe that the set of κ-continuous functions from $P(|\mathcal{M}|)^n$ to $P(|\mathcal{M}|)$, with $n \geqslant 1$, contains constants, projections, intersections and unions, as well as the unary functions $\phi_{\mathcal{M}}$ with $\phi = \langle a \rangle x$ for some $a \in \mathit{Act}$. Moreover, this set is closed under composition and diagonalisation, and so Propositions 6 and 7 immediately yield the following result:

▶ **Proposition 11.** *Every formula in the fragment $\mathcal{C}_{\aleph_1}(X)$ is $\aleph_1$-continuous in X.*

4.1 Syntactic considerations

▶ **Definition 12.** The *digraph* $G(\phi)$ of a formula $\phi \in \mathsf{L}_\mu$ is obtained from the syntax tree of ϕ by adding an edge from each occurrence of a bound variable to its binding fixed-point quantifier. The root of $G(\phi)$ is ϕ.

▶ **Definition 13.** A path in $G(\phi)$ is *bad* if one of its nodes corresponds to a subformula occurrence of the form $[a]\psi$. A bad cycle in $G(\phi)$ is a bad path starting and ending at the same vertex.

Recall that a path in a digraph is simple if it does not visit twice the same vertex. The rooted digraph $G(\phi)$ is a tree with back-edges; in particular, it has this property: for every node, there exists a unique simple path from the root to this node.

▶ **Definition 14.** We say that an occurrence of a free variable x of ϕ is

1. *bad* if there is a bad path in $G(\phi)$ from the root to it;
2. *not-so-bad* (or *boxed*) if the unique simple path in $G(\phi)$ from the root to it is bad;
3. *very bad* if it is bad and not boxed.

▶ **Example 15.** Figure 1 represents the digraph of the formula

$$\mu_{z_1}.(y_0 \wedge \nu_{z_0}.(z_0 \wedge [\ ]z_1)) \vee (\langle\ \rangle y_0 \wedge y_1).$$

From the figure we observe that:

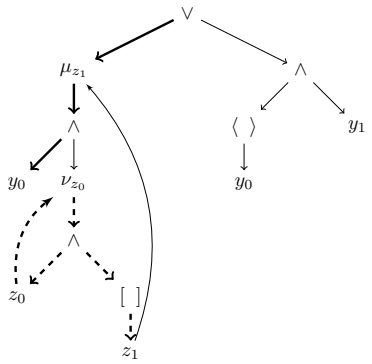


Figure 1 The digraph of a formula in L_μ.

- The free occurrence of z_1 in the digraph of $\nu_{z_0}.(z_0 \wedge [\ ]z_1)$ (in dashed) is bad but not-so-bad.
- The free occurrence of y_0 in the left branch of the digraph (in bold) is very bad. The other occurrence of y_0 is not bad.
- The unique free occurrence of y_1 in ϕ is not bad.

▶ **Lemma 16.** *For every set X of variables and every $\phi \in \mathsf{L}_\mu$, the following are equivalent:*

1. $\phi \in \mathcal{C}_{\aleph_1}(X)$,
2. *no occurrence of a variable $x \in X$ is bad in ϕ.*

4.2 The $\mathcal{C}_{\aleph_1}(x)$-flattening of formulas

We aim at defining the $\mathcal{C}_{\aleph_1}(x)$-flattening $\phi^{\flat x}$ of any formula ϕ of the modal μ-calculus. This will go through the definition of the intermediate formula $\phi^{\sharp x}$ which has one more new free variable $\overline{x}$. The formula $\phi^{\sharp x}$ is obtained from ϕ by renaming to $\overline{x}$ all the boxed occurrences of the variable x. The formal definition is given by induction as follows:

$$
\begin{array}{cccc}
y^{\sharp x} = y & (\neg y)^{\sharp x} = \neg y & \top^{\sharp x} = \top & \bot^{\sharp x} = \bot \\
(\psi_0 @ \psi_1)^{\sharp x} = \psi_0^{\sharp x} @ \psi_1^{\sharp x} & \text{with } @ \in \{\wedge, \vee\}, & (\langle a\rangle\psi)^{\sharp x} = \langle a\rangle\psi^{\sharp x} & ([a]\psi)^{\sharp x} = [a]\psi[\overline{x}/x] \\
(Q_z.\psi)^{\sharp x} = Q_z.\psi^{\sharp x} & \text{with } Q \in \{\mu, \nu\}. & &
\end{array}
$$

In the definition of $\phi^{\sharp x}$ above, we assume that x has no bound occurrences in ϕ. The following fact is proved by a straightforward induction.

▶ **Lemma 17.** *Let $\phi \in \mathsf{L}_\mu$. We have $\phi^{\sharp x} \cdot [x/\overline{x}] = \phi$.*

The $\mathcal{C}_{\aleph_1}(x)$-flattening $\phi^{\flat x}$ of formula $\phi \in \mathsf{L}_\mu$ is then defined by:

$$\phi^{\flat x} := \phi^{\sharp x} \cdot [\bot/\overline{x}]$$

and henceforward we shorten it up to $\phi^\flat$.

Let us notice that $\phi^{\sharp x}$ (or $\phi^\flat$) does not in general belong to $\mathcal{C}_{\aleph_1}(x)$. For example, $(\mu_z.x \vee [a]z)^\flat = \mu_z.x \vee [a]z \notin \mathcal{C}_{\aleph_1}(x)$ since $x \vee [a]z \notin \mathcal{C}_{\aleph_1}(\{x, z\})$. Yet, the following definition and lemma partially justify the choice of naming.

▶ **Definition 18.** A formula ϕ is *almost-good* w.r.t. a set X of variables if no occurrence of a variable $x \in X$ is very bad. A formula ϕ is *almost-good* if it is almost-good w.r.t. $\{ x \}$.

▶ **Lemma 19.** *If ϕ is an almost-good formula, then both $\phi^{\sharp x}$ and $\phi^\flat$ belong to $\mathcal{C}_{\aleph_1}(x)$.*

We aim therefore to transform a formula ϕ into an equivalent formula in which there are not very bad occurrences of the variable x. The transformation that we define next achieves this goal. For $\phi \in \mathsf{L}_\mu$ and a finite set X of variables not bound in ϕ, we define $\psi^{\Box X}$ as follows. When in ψ no occurrence of a variable $x \in X$ is very bad, we take $\psi^{\Box X} := \psi$. Otherwise:

$$(\langle a \rangle \psi)^{\Box X} := \langle a \rangle (\psi)^{\Box X}, \qquad (\psi_1 @ \psi_2)^{\Box X} := (\psi_1)^{\Box X} @ (\psi_2)^{\Box X}, \quad \text{with } @ \in \{ \wedge, \vee \},$$
$$(Q_z.\psi)^{\Box X} := \psi_0[\psi_1/\overline{z}], \text{ where } \psi_0 := Q_z.\psi_2,\ \psi_1 := Q_{\overline{z}}.\psi_0,\ \psi_2 := (\psi^{\Box X \cup \{z\}})^{\sharp z},$$

with $Q \in \{ \mu, \nu \}$. That is, in the last clause, ψ_2 is obtained from $\psi^{\Box X \cup \{z\}}$ by renaming all the boxed occurrences of z to $\overline{z}$. Observe that the first defining clause implies that

$$x^{\Box X} = x \text{ if } x \in X, \quad \psi^{\Box X} = \psi \text{ if } \psi \text{ contains no variable } x \in X, \quad \text{and } ([a]\psi)^{\Box X} = [a]\psi.$$

▶ **Proposition 20.** *The formula $\phi^{\Box X}$ is almost-good w.r.t. X and it is equivalent to ϕ.*

We can finally state the main result up to now.

▶ **Theorem 21.** *Every formula ϕ is equivalent to a formula ψ with $\psi^{\sharp x}$ and $\psi^\flat$ in $\mathcal{C}_{\aleph_1}(x)$.*

4.3 Comparing the closures of ϕ and $\phi^\flat$

We develop here some syntactic considerations allowing us to relate the closures of ϕ and $\phi^\flat$. In turn, this will make it possible to relate the positions of the games $\mathcal{G}(\mathcal{M}, \phi)$ and $\mathcal{G}(\mathcal{M}, \phi^\flat)$, so to construct, in the proof of Proposition 24, a winning strategy in the latter game from a winning strategy in the former.

▶ **Lemma 22.** *If $\overline{x}$ is a free variable of ϕ and κ is either a variable not bound in ϕ or a constant, then*

$$\mathrm{CL}(\phi \cdot [\kappa/\overline{x}]) = \{ \psi \cdot [\kappa/\overline{x}] \mid \psi \in \mathrm{CL}(\phi) \}.$$

In particular, we have

$$\mathrm{CL}(\phi) = \{ \phi' \cdot [x/\overline{x}] \mid \phi' \in \mathrm{CL}(\phi^{\sharp x}) \}, \quad \mathrm{CL}(\phi^\flat) = \{ \phi' \cdot [\bot/\overline{x}] \mid \phi' \in \mathrm{CL}(\phi^{\sharp x}) \}.$$

The second statement of the lemma is an immediate of the first, considering that $\phi = \phi^{\sharp x} \cdot [x/\overline{x}]$ and $\phi^\flat = \phi^{\sharp x} \cdot [\bot/\overline{x}]$.

4.4 The continuous fragments

Our next goal is to prove some sort of converse to Proposition 11.

A pointed model $\langle \mathcal{M}, s \rangle$ is a *tree model* if the rooted digraph $\langle |\mathcal{M}|, \bigcup_{a \in Act} R_a, s \rangle$ is a tree. Let κ be a cardinal. A tree model $\langle \mathcal{M}, s \rangle$ is *κ-expanded* if, for each $a \in Act$, whenever xR_ax', there are at least κ a-successors of x that are bisimilar to x'. The following lemma is a straightforward generalization of [10, Proposition 1].

▶ **Lemma 23.** *For each pointed model $\langle \mathcal{M}, s \rangle$ there exists a κ-expanded tree model $\langle \mathcal{T}, t \rangle$ bisimilar to $\langle \mathcal{M}, s \rangle$.*

▶ **Proposition 24.** *If $\mathcal{M}, s \Vdash \phi$ and ϕ is κ-continuous in x, then $\mathcal{M}, s \Vdash \phi^\flat$.*

Proof. Suppose that $\mathcal{M} = (|\mathcal{M}|, \{ R_a \mid a \in A \}, v)$ is a model and that $s_0 \Vdash \phi$. We want to prove that $s_0 \Vdash \phi^\flat$. Notice first that, by Lemma 23, we can assume that $\langle \mathcal{M}, s_0 \rangle$ is κ-expanded tree model.

Since ϕ is κ-continuous in x and $s_0 \in \phi_{\mathcal{M}}(v(x))$, there exists $U \subseteq v(x)$, with cardinality of U striclty smaller than κ, such that $s_0 \in \phi_{\mathcal{M}}(U)$, so $\mathcal{M}[U/x], s_0 \Vdash \phi$. We shall argue that $\mathcal{M}[U/x], s_0 \Vdash \phi^\flat$, from which it follows that $s_0 \in \phi^\flat_{\mathcal{M}}(U) \subseteq \phi^\flat_{\mathcal{M}}(v(x))$—since $\phi^\flat_{\mathcal{M}}$ is monotonic—thus $\mathcal{M}, s_0 \Vdash \phi^\flat$.

In the following let $\mathcal{N} = \mathcal{M}[U/x]$ (notice that $\mathcal{N}$ is not anymore κ-expanded). Since $\mathcal{N}, s_0 \Vdash \phi$, let us fix a winning strategy for Eva in the game $\mathcal{G}(\mathcal{N}, \phi)$ from position (s_0, ϕ). We define next a strategy for Eva in the game $\mathcal{G}(\mathcal{N}, \phi^\flat)$ from position $(s_0, \phi^\flat)$. Observe first that, by Lemma 22, positions in $\mathcal{G}(\mathcal{N}, \phi)$ (respectively, $\mathcal{G}(\mathcal{N}, \phi^\flat)$) are of the form $(s, \psi[x/\overline{x}])$ (resp., $(s, \psi[\bot/\overline{x}])$) for a formula $\psi \in \mathrm{CL}(\phi^{\sharp x})$. Therefore, at the beginning of the play, Eva plays in $\mathcal{G}(\mathcal{N}, \phi^\flat)$ simulating the moves of the given winning strategy for the game $\mathcal{G}(\mathcal{N}, \phi)$. The simulation goes on until the play reaches a pair of positions $p = (s, [a]\chi\sigma^{\phi^{\sharp x}}_{[a]\chi} \cdot [x/\overline{x}])$ and $p' = (s, [a]\chi\sigma^{\phi^{\sharp x}}_{[a]\chi} \cdot [\bot/\overline{x}])$, for some subformula $[a]\chi$ of $\phi^{\sharp x}$, where $\chi = \chi'[\overline{x}/x]$ for some subformula χ' of ϕ.

▶ **Claim.** *The positions p and p' are respectively of the form $(s, [a]\psi) \in \mathcal{G}(\mathcal{N}, \phi)$ and $(s, [a]\psi') \in \mathcal{G}(\mathcal{N}, \phi^\flat)$ for some ψ and ψ' such that $\psi[\bot/x] \to \psi'$ is a tautology.*

Thus Eva needs to continue playing in the game $\mathcal{G}(\mathcal{N}, \phi^\flat)$ from a position of the form $(s, [a]\psi')$ where $\psi[\bot/x] \to \psi'$ is a tautology. We construct a winning stategy for Eva from this position as follows. Since the play has reached the position $(s, [a]\psi)$ of $\mathcal{G}(\mathcal{N}, \phi)$ we also know that $s \in [\![[a]\psi]\!]_{\mathcal{N}}$. We argue then that $s \in [\![[a]\psi]\!]_{\mathcal{N}}$ implies $s \in [\![[a]\psi[\bot/x]]\!]_{\mathcal{N}}$. Since $[\![[a]\psi[\bot/x]]\!]_{\mathcal{N}} \subseteq [\![[a]\psi']\!]_{\mathcal{N}}$, Eva also has a winning strategy from position $(s, [a]\psi')$ of the game $\mathcal{G}(\mathcal{N}, \phi^\flat)$, which she shall use to continue the play.

▶ **Claim.** *$s \in [\![[a]\psi]\!]_{\mathcal{N}}$ implies $s \in [\![[a]\psi[\bot/x]]\!]_{\mathcal{N}}$.*

Proof of Claim. The statement of the claim trivially holds if s has no successors. Let s' be a fixed a-successor of s (i.e. sR_as'), so $\mathcal{N}, s' \Vdash \psi$; we want to show that $\mathcal{N}, s' \Vdash \psi[\bot/x]$. To this goal, recalling that $\psi[\bot/x] \in \mathsf{L}_\mu[Prop \setminus \{x\}, Act]$ and using Proposition 9, it is enough to prove that $\langle \mathcal{N}, s' \rangle$ is $(Prop \setminus \{x\}, Act)$-bisimilar to some $\langle \mathcal{N}, s'' \rangle$ such that $\mathcal{N}, s'' \Vdash \psi[\bot/x]$.

Let S be the set

$$\{ t \mid sR_at,\ \langle \mathcal{M}, t \rangle \text{ is bisimilar to } \langle \mathcal{M}, s' \rangle,\ \text{and } {\downarrow}t \cap U \neq \emptyset \},$$

where we have used ${\downarrow}t$ to denote the subtree of $\langle \mathcal{M}, s_0 \rangle$ rooted at t. Recall that the cardinality of U is strictly smaller than κ and so is the cardinality of S once it is at most equal to the cardinality of U. But the cardinality of $\{ t \mid sR_at,\ \langle \mathcal{M}, t \rangle$ is bisimilar to $\langle \mathcal{M}, s' \rangle \}$ is at least κ (recall $\langle \mathcal{M}, s_0 \rangle$ is a κ-expanded tree model). Consequently there must be a successor s'' of s such that $\langle \mathcal{M}, s'' \rangle$ is bisimilar to $\langle \mathcal{M}, s' \rangle$ and which does not belong to S, that is ${\downarrow}s'' \cap U = \emptyset$ (i.e. no states in U are reachable from s''). Since $\mathcal{N}, s'' \Vdash \psi$ and ${\downarrow}s'' \cap U = \emptyset$, we have $\mathcal{N}, s'' \Vdash \psi[\bot/x]$. Yet $\langle \mathcal{M}, s'' \rangle$ and $\langle \mathcal{M}, s' \rangle$ are bisimilar and since $\mathcal{N}$ is obtained from $\mathcal{M}$ just by modifying the value of the variable x, $\langle \mathcal{N}, s'' \rangle$ and $\langle \mathcal{N}, s' \rangle$ are $(Prop \setminus \{ x \}, Act)$-bisimilar. As stated before, this together with $\mathcal{N}, s'' \Vdash \psi[\bot/x]$ imply that $\mathcal{N}, s' \Vdash \psi[\bot/x]$. ◀

To complete the proof of Proposition 24 we need to argue that the strategy so defined for Eva to play in the game $\mathcal{G}(\mathcal{M}, \phi^\flat)$ is winning. The only difficulty in asserting this is to exclude

the case where the initial simulation leads to a pair of positions of the form $(s, \overline{x}[x/\overline{x}])$ and $(s, \overline{x}[\bot/\overline{x}])$. This is however excluded since in $\phi^{\sharp x}$ all the occurrences of $\overline{x}$ are boxed, so we are enforced to go through the second step of the strategy. ◀

▶ **Proposition 25.** *If, for some regular cardinal κ, $\phi \in \mathsf{L}_\mu$ is κ-continuous, then ϕ is equivalent to $\phi^\flat$.*

Proof. Notice that, by monotonicity in the variable x, $\phi^\flat \to \phi$ is a tautology. Proposition 24 exhibits the converse implication as another tautology. ◀

▶ **Theorem 26.** *If for some regular cardinal κ, $\phi \in \mathsf{L}_\mu$ is a κ-continuous formula, then ϕ is equivalent to a formula $\phi' \in \mathcal{C}_{\aleph_1}(x)$.*

Proof. Suppose that ϕ is κ-continuous. By Corollary 21, ϕ is equivalent to a formula ψ with $\psi^\flat \in \mathcal{C}_{\aleph_1}(x)$. Clearly, ψ is κ-continuous as well, so it is equivalent to $\psi^\flat$ by Proposition 25. It follows that ϕ is equivalent to $\psi^\flat \in \mathcal{C}_{\aleph_1}(x)$. ◀

As a consequence of the previous Theorem 26, we obtain the following result.

▶ **Theorem 27.** *There are only two fragments of the modal μ-calculus determined by continuity conditions: the fragment $\mathcal{C}_{\aleph_0}(x)$ and the fragment $\mathcal{C}_{\aleph_1}(x)$.*

▶ **Theorem 28.** *The following problem is decidable: given a formula $\phi(x) \in \mathsf{L}_\mu$, is $\phi(x)$ κ-continuous for some regular cardinal κ?*

Proof. From what has been exposed above, ϕ is κ-continuous if and only if it equivalent to the formula $\phi' \in \mathcal{C}_{\aleph_1}(x)$, where $\phi' = (\phi^{\Box x})^\flat$. It is then enough to observe that there are effective processes to construct the formula ϕ' and to check whether ϕ is equivalent to ϕ'. ◀

5 Large closure ordinals

We start by presenting some of the tools required for the two subsections in which this section is organized. Then, we prove that ω_1, the least uncountable ordinal, is a closure ordinal for the modal μ-calculus. Finally, in the second subsection, we show that the set of closure ordinals is closed under the ordinal sum.

▶ **Definition 29.** Let $\phi(x)$ be a formula of the modal μ-calculus. We say that an ordinal α is the *closure ordinal* of ϕ (and write $\mathsf{cl}(\phi) = \alpha$) if, for each model $\mathcal{M}$, the function $\phi_\mathcal{M}$ converges to its least fixed-point in at most α steps, and there exists a model $\mathcal{M}$ in which $\phi_\mathcal{M}$ converges to its least fixed-point in exactly α steps.

▶ **Lemma 30.** *If α is a closure ordinal, then there is a formula $\phi(x)$ such that $\mathsf{cl}(\phi(x)) = \alpha$ and that is* total*, meaning that $[\![\mu_x.\phi(x)]\!]_\mathcal{M} = |\mathcal{M}|$, for each model $\mathcal{M}$.*

▶ **Proposition 31.** *If a formula $\phi(x)$ belongs to the syntactic fragment $\mathcal{C}_{\aleph_1}(x)$, then it has a closure ordinal $\mathsf{cl}(\phi(x))$ and ω_1 is an upper bound for $\mathsf{cl}(\phi(x))$.*

Proof. The formula ϕ belongs to the syntactic fragment $\mathcal{C}_{\aleph_1}(x)$, thus it is $\aleph_1$-continuous and, for every model $\mathcal{M}$, $\phi_\mathcal{M}$ is $\aleph_1$-continuous. It follows then from Proposition 5 that $\phi_\mathcal{M}$ converges to its least fixed-point in at most ω_1 steps. ◀

5.1 ω_1 is a closure ordinal

We are going to prove that ω_1 is the closure ordinal of the following bimodal formula:

$$\Phi(x) := \nu_z.(\langle v\rangle x \wedge \langle h\rangle z) \vee [v]\bot\,. \tag{1}$$

Later we shall also argue that ω_1 is the closure ordinal of a *monomodal* formula.

For the time being, consider $Act = \{h, v\}$; if $\mathcal{M} = \langle |\mathcal{M}|, R_h, R_v, v\rangle$ is a model, we think of R_h as a set of horizontal transitions and of R_v as a set of vertical transitions. Thus, for $s \in |\mathcal{M}|$, $\mathcal{M}, s \Vdash \Phi(x)$ if either (i) there are no vertical transitions from s, or (ii) there exists an infinite horizontal path from s such that each state on this path has a vertical transition to a state s' such that $\mathcal{M}, s' \Vdash x$.

By Proposition 31, the formula $\Phi(x)$ has a closure ordinal and $\mathsf{cl}(\Phi(x)) \leqslant \omega_1$. In order to prove that $\mathsf{cl}(\Phi(x)) = \omega_1$, we are going to construct a model $\mathcal{M}_{\omega_1}$ where $\Phi^{\omega_1}_{\mathcal{M}_{\omega_1}}(\emptyset) \not\subseteq \Phi^{\alpha}_{\mathcal{M}_{\omega_1}}(\emptyset)$ for each $\alpha < \omega_1$.

The construction relies on few combinatorial properties of posets and ordinals that we recall here. For a poset P and an ordinal α, an α-chain in P is a subset $\{\, p_\beta \mid \beta < \alpha \,\} \subseteq P$, with $p_\beta \leq p_\gamma$ whenever $\beta \leq \gamma < \alpha$. An α-chain $\{\, p_\beta \mid \beta < \alpha \,\} \subseteq P$ is *cofinal* in P if, for every $p \in P$ there exists $\beta < \alpha$ with $p \leq p_\beta$. The *cofinality* κ_P of a poset P is the least ordinal α for which there exists an α-chain cofinal in P. Recall that an ordinal α might be identified with the poset $\{\, \beta \mid \beta \text{ is an ordinal}, \beta < \alpha \,\}$ and so $\kappa_\alpha = \omega$, whenever α is a countable infinite limit ordinal; this means that, for such an α, it is always possible to pick an ω-chain cofinal in α.

For a given ordinal $\alpha \leq \omega_1$, let

$$S_\alpha := \{\, (\beta, n) \mid \beta \text{ is an ordinal}, \beta < \alpha, 0 \leq n < \omega \,\}\,.$$

We define $\mathcal{M}_{\omega_1}$ to be the model $\langle S_{\omega_1}, R_h, R_v, v\rangle$ where $v(y) = \emptyset$, for each $y \in Prop$, horizontal transitions are of the form $(\beta, n)R_h(\beta, n+1)$, for each ordinal β and each $n < \omega$, and vertical transitions from a state $(\beta, n) \in S_{\omega_1}$ are as folllows:

- if $\beta = 0$, then there are no vertical transitions outgoing from $(0, n)$;
- if $\beta = \gamma + 1$ is a successor ordinal, then the only vertical transitions are of the form $(\gamma + 1, n)R_v(\gamma, 0)$;
- for β a countable limit ordinal distinct from 0, the vertical transitions are of the form $(\beta, n)R_v(\beta_n, 0)$, where the set $\{\, \beta_n \mid n < \omega \,\}$ is an ω-chain cofinal in β.

We prove that, we have $\Phi_{\mathcal{M}_{\omega_1}}(S_\alpha) = S_{\alpha+1}$, for each countable ordinal α, and, consequently, $\Phi^{\alpha}_{\mathcal{M}_{\omega_1}}(\emptyset) = S_\alpha$, for each ordinal $\alpha \leq \omega_1$. To conclude the proof, it is enough to observe that $S_{\omega_1} \not\subseteq S_\alpha$, for each $\alpha < \omega_1$. Indeed, if $\alpha < \omega_1$, then we can find an ordinal β with $\alpha < \beta < \omega_1$, so the states (β, n), $n \geq 0$, do not belong to S_α.

▶ **Theorem 32.** *The closure ordinal of $\Phi(x)$ is ω_1.*

5.2 From a bimodal language to a monomodal language

The following statement generalizes to the modal μ-calculus a well known coding of polymodal logic to monomodal logic, see [29] and [18, Section 4].

▶ **Proposition 33.** *For each bimodal formula ϕ of the modal μ-calculus, we construct a monomodal formula $\phi^{\mathtt{sim}}$; if ϕ belongs to $\mathcal{C}_{\aleph_1}(x)$, then so does $\phi^{\mathtt{sim}}$. Moreover, for each bimodal model $\mathcal{M}$ we can also construct a monomodal model $\mathcal{M}^{sim}$, together with an injective function $(-)^\circ : |\mathcal{M}| \to |\mathcal{M}^{sim}|$ such that, for each $s \in |\mathcal{M}|$, $\mathcal{M}, s \Vdash \phi$ if and only if $M^{sim}, s^\circ \Vdash \phi^{\mathtt{sim}}$.*

▶ **Theorem 34.** *The monomodal formula $\Phi^{\mathtt{sim}}$ has closure ordinal ω_1.*

Proof. Since the translation $\phi \mapsto \phi^{\mathtt{sim}}$ sends formulas in $\mathcal{C}_{\aleph_1}(x)$ to formulas in $\mathcal{C}_{\aleph_1}(x)$, $\Phi^{\mathtt{sim}}$ is $\aleph_1$-continuous and therefore it has a closure ordinal bounded by ω_1. To argue that the closure ordinal of $\Phi^{\mathtt{sim}}$ is equal to ω_1 it is enough to consider the model $\mathcal{M}^{sim}_{\omega_1}$ and rely on Proposition 33. ◀

5.3 Closure under ordinal sum

Here we prove that the sum of any two closure ordinals is again a closure ordinal. To ease the exposition, we shall make use of the universal modality $[\mathcal{U}]$ of the μ-calculus which, in case of a monomodal language, is defined as $[\mathcal{U}]\chi := \nu_z.(\,\chi \wedge [\ ]z\,)$. The modal operator $[\mathcal{U}]$ does not satisfy the Euclidean axiom **5**, yet it is satisfies all the axioms of an **S4** modality.

▶ **Theorem 35.** *Suppose $\phi_0(x)$ and $\phi_1(x)$ are monomodal formulas that have, respectively, α and β as closure ordinals. For a variable p occurring neither in ϕ_0 nor in ϕ_1, for $\chi := \chi_0 \wedge \chi_1$ with $\chi_0 = \neg p \to ([\ ]\neg p \wedge (\neg p \wedge \mu_z.\phi_0(z)))$ and $\chi_1 := p \to ([\ ](\neg p \to \mu_z.\phi_0(z)) \wedge \mu_z.\mathtt{tr}(\phi_1(z)))$, and for*

$$\psi(x) := (\,\neg p \wedge \phi_0(x)\,) \vee (\,\mathtt{tr}(\phi_1)(x) \wedge [\ ](\neg p \to x)\,),$$

the formula $\Psi(x) := [\mathcal{U}]\chi \wedge \psi(x)$ has closure ordinal $\alpha + \beta$.

We prove the theorem through a series of observations. Say that a model $\mathcal{N}$ is *acceptable* if $\mathcal{N} \models [\mathcal{U}]\chi$. The first observation is the following: *an ordinal γ is the closure ordinal of the formula $\Psi(x)$ if and only if* (i) *the formula $\psi(x)$ converges to its least fixed point in at most γ steps on all the acceptable models, and* (ii) *there exists an acceptable model on which the formula $\psi(x)$ converges to its least fixed point in exactly γ steps.*

We continue by understanding how $\psi_{\mathcal{N}}$ acts on an acceptable model $\mathcal{N}$. To this goal, let $\mathcal{N}_0$ and $\mathcal{N}_1$ be the submodels of $\mathcal{N}$ induced by $v(\neg p)$ and $v(p)$, respectively. To ease the reading, let also $N_0 := v(\neg p)$, $N_1 := v(p)$, $\phi_{\mathcal{N}_0} := \phi_{0\mathcal{N}_0}$ and $\phi_{\mathcal{N}_1} := \phi_{1\mathcal{N}_1}$, so $\phi_{\mathcal{N}_0} : P(N_0) \to P(N_0)$ and $\phi_{\mathcal{N}_1} : P(N_1) \to P(N_1)$. Observe that since $\mathcal{N}, s \models \neg p \to [\ ]\neg p$ for every $s \in |\mathcal{N}|$, N_0 is a closed subset of $|\mathcal{N}|$. Then, by Proposition 10, we have $\psi_{\mathcal{N}}(X) \cap N_0 = \phi_{\mathcal{N}_0}(X \cap N_0)$ and $\mathtt{tr}(\phi_1)_{\mathcal{N}_1}(X) = \phi_{\mathcal{N}_1}(X \cap N_1)$ for each $X \subseteq |\mathcal{N}|$. Now let $\nabla(X) := N_1 \cap [\ ]_{\mathcal{N}}(N_0 \to X)$. We consider that the domain of ∇ is $P(N_0)$ while its codomain is $P(N_1)$. Therefore, $\psi_{\mathcal{N}}$ is of the form

$$\psi_{\mathcal{N}}(X) = \phi_{\mathcal{N}_0}(X \cap N_0) \cup (\phi_{\mathcal{N}_1}(X \cap N_1) \cap \nabla(X \cap N_0))\,. \tag{2}$$

We notice that if $\mathcal{N}$ is an acceptable model, then $N_0 = [\![\mu_z.\phi_0(z)]\!]_{\mathcal{M}} = \phi^{\alpha}_{\mathcal{N}_0}(\emptyset)$ and $N_1 = [\![\mu_z.\phi_1(z)]\!]_{\mathcal{M}} = \phi^{\beta}_{\mathcal{N}_1}(\emptyset)$. Moreover, $\mathcal{N}, s \models p \to [\ ](\neg p \to \mu_x.\phi_0(x))$, for each $s \in |\mathcal{N}|$, so

$$\nabla(X) = N_1, \quad \text{whenever } X \supseteq \phi^{\alpha}_{\mathcal{N}_0}(\emptyset)(= N_0). \tag{3}$$

▶ **Proposition 36.** *On every acceptable model $\mathcal{N}$ the equality $\psi^{\alpha+\beta}_{\mathcal{N}}(\emptyset) = |\mathcal{N}|$ holds and, consequently, the formula $\psi(x)$ converges before $\alpha + \beta$ steps.*

Proof. Since N_0 is a closed subset of $|\mathcal{N}|$, by Proposition 10, we have

$$\psi^{\delta}_{\mathcal{N}}(\emptyset) \cap N_0 = \psi^{\delta}_{\mathcal{N}_0}(\emptyset) = \phi^{\delta}_{\mathcal{N}_0}(\emptyset) \tag{4}$$

for each ordinal δ. Consequently, $\psi^{\alpha+\gamma}_{\mathcal{N}}(\emptyset) \cap N_0 \supseteq \psi^{\alpha}_{\mathcal{N}}(\emptyset) \cap N_0 = \phi^{\alpha}_{\mathcal{N}_0}(\emptyset)$, for every ordinal γ. By a straightforward induction we also prove that, for each ordinal γ,

$$\phi^{\gamma}_{\mathcal{N}_1}(\emptyset) \subseteq \psi^{\alpha+\gamma}_{\mathcal{N}}(\emptyset) \cap N_1 \,. \tag{5}$$

Therefore $|\mathcal{N}| = N_0 \cup N_1 = \phi^{\alpha}_{\mathcal{N}_0}(\emptyset) \cup \phi^{\beta}_{\mathcal{N}_1}(\emptyset) \subseteq (\psi^{\alpha+\beta}_{\mathcal{N}}(\emptyset) \cap N_0) \cup (\psi^{\alpha+\beta}_{\mathcal{N}}(\emptyset) \cap N_1) = \psi^{\alpha+\beta}_{\mathcal{N}}(\emptyset)$. ◀

▶ **Proposition 37.** *There exists an acceptable model $\mathcal{N}$ on which $\psi(x)$ converges exactly after $\alpha + \beta$ steps.*

Proof. Since the formulas $\phi_0(x)$ and $\phi_1(x)$ have, respectively, α and β as closure ordinals, by Proposition 30 there exist models $\mathcal{M}_\gamma = \langle |\mathcal{M}_\gamma|, R_\gamma, v_\gamma \rangle$, $\gamma \in \{\, \alpha, \beta \,\}$, such that for every $\alpha' < \alpha$ and $\beta' < \beta$ $[\![\mu_x.\phi_0(x)]\!]_{\mathcal{M}_\alpha} = |\mathcal{M}_\alpha| = \phi_0{}^{\alpha}_{\mathcal{M}_\alpha}(\emptyset) \neq \phi_0{}^{\alpha'}_{\mathcal{M}_\alpha}(\emptyset)$ and $[\![\mu_x.\phi_1(x)]\!]_{\mathcal{M}_\beta} = |\mathcal{M}_\beta| = \phi_1{}^{\beta}_{\mathcal{M}_\beta}(\emptyset) \neq \phi_1{}^{\beta'}_{\mathcal{M}_\beta}(\emptyset)$.

We construct now the model $\mathcal{M}_{\alpha+\beta}$ by making the disjoint union of the sets $|\mathcal{M}_\alpha|$ and $|\mathcal{M}_\beta|$, endowed with $R_\alpha \cup R_\beta \cup \{\, (s, s') \mid s \in |\mathcal{M}_\beta|, s' \in |\mathcal{M}_\alpha| \,\}$ and the valuation v defined by $v(q) := |\mathcal{M}|_\beta$, if $q = p$, and $v(q) := v_\alpha(q) \cup v_\beta(q)$ otherwise. Let us put $\mathcal{N} = \mathcal{M}_{\alpha+\beta}$. Observe now that $\mathcal{M}_{\alpha+\beta}$ is an acceptable model and that $\nabla(X) = \emptyset$ for every $X \subseteq |\mathcal{N}|$ such that $X \cap N_0 \subsetneq \phi^{\alpha}_{\mathcal{N}_0}(\emptyset)$. Because of this, the inclusion (5) is actually an equality. But then we apply equations (4) and (5) to obtain $\psi^{\alpha}_{\mathcal{N}}(\emptyset) = \phi^{\alpha}_{\mathcal{N}_0}(\emptyset) \neq \phi^{\delta}_{\mathcal{N}_0}(\emptyset) = \psi^{\delta}_{\mathcal{N}}(\emptyset)$ and $\psi^{\alpha+\gamma}_{\mathcal{N}}(\emptyset) = N_0 \cup \phi^{\gamma}_{\mathcal{N}_1}(\emptyset)$, for ordinals $\delta < \alpha$ and γ. Finally, $\psi^{\alpha+\beta}_{\mathcal{N}}(\emptyset) = |\mathcal{N}| = N_0 \cup \phi^{\beta}_{\mathcal{N}_1}(\emptyset) \neq N_0 \cup \phi^{\gamma}_{\mathcal{N}_1}(\emptyset) = \psi^{\alpha+\gamma}_{\mathcal{N}}(\emptyset)$, for $\gamma < \beta$. ◀

References

1. J. Adamek and J. Rosicky. *Locally Presentable and Accessible Categories:*. Cambridge University Press, Cambridge, 1994.
2. B. Afshari and G. E. Leigh. On closure ordinals for the modal mu-calculus. In Ronchi Della Rocca [23], pages 30–44. URL: `http://www.dagstuhl.de/dagpub/978-3-939897-60-6`, `doi:10.4230/LIPIcs.CSL.2013.30`.
3. L. Alberucci and A. Facchini. The Modal μ-Calculus Hierarchy on Restricted Classes of Transition Systems. *The Journal of Symbolic Logic*, 74(4):1367–1400, December 2009. URL: `https://hal.archives-ouvertes.fr/hal-00396431`.
4. D. Berwanger, E. Grädel, and G. Lenzi. On the variable hierarchy of the modal μ-calculus. In J. Bradfield, editor, *CSL 2002, Proceedings*, pages 352–366, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. `doi:10.1007/3-540-45793-3_24`.
5. M. Bojanczyk, C. Dittmann, and S. Kreutzer. Decomposition theorems and model-checking for the modal μ-calculus. In *CSL-LICS'14*, pages 17:1–17:10, New York, NY, USA, 2014. ACM. `doi:10.1145/2603088.2603144`.
6. J. Bradfield and I. Walukiewicz. The mu-calculus and model-checking. In E. Clarke, T. Henzinger, and H. Veith, editors, *Handbook of Model Checking*. Springer-Verlag, 2015.
7. J. C. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195(2):133–153, 1998. `doi:10.1016/S0304-3975(97)00217-X`.
8. R. Cousot and P. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43–57, 1979.
9. M. Czarnecki. How fast can the fixpoints in modal μ-calculus be reached? In L. Santocanale, editor, *7th Workshop on Fixed Points in Computer Science, FICS 2010*, page 89, Brno, Czech Republic, August 2010. Available from `Hal`: `https://hal.archives-ouvertes.fr/hal-00512377`.
10. G. Fontaine. Continuous fragment of the mu-calculus. In M. Kaminski and S. Martini, editors, *CSL 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 139–153. Springer, 2008. `doi:10.1007/978-3-540-87531-4_12`.

11 J. Fortier and L. Santocanale. Cuts for circular proofs: semantics and cut-elimination. In Ronchi Della Rocca [23], pages 248–262. URL: http://www.dagstuhl.de/dagpub/978-3-939897-60-6, doi:10.4230/LIPIcs.CSL.2013.248.

12 V. Vaccaro G. Sambin. A new proof of Sahlqvist's theorem on modal definability and completeness. *Journal of Symbolic Logic*, 54:992–999, 1989.

13 M. J. Gouveia and L. Santocanale. $\aleph_1$ and the modal μ-calculus. Preprint, available from Hal: https://hal.archives-ouvertes.fr/hal-01503091, March 2017.

14 D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *CONCUR'96, Proceedings*, pages 263–277, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. doi:10.1007/3-540-61604-7_60.

15 M. Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, 1998. doi:10.1016/S0020-0190(98)00150-1.

16 B. Jónsson. On the canonicity of sahlqvist identities. *Studia Logica*, 53(4):473–491, 1994. URL: http://www.jstor.org/stable/20015747.

17 D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.

18 M. Kracht and F. Wolter. Simulation and transfer results in modal logic – A survey. *Studia Logica*, 59(1):149–177, 1997. doi:10.1023/A:1004900300438.

19 J.-L. Lassez, V. L. Nguyen, and L. Sonenberg. Fixed point theorems and semantics: A folk tale. *Inf. Process. Lett.*, 14(3):112–116, 1982. doi:10.1016/0020-0190(82)90065-5.

20 G. Lenzi. The modal μ-calculus: a survey. *Task Quarterly*, 9(3):29–316, 2005.

21 D. Niwiński and I. Walukiewicz. Games for the μ-calculus. *Theoretical Computer Science*, 163(1):99–116, 1996. doi:10.1016/0304-3975(95)00136-0.

22 J. Obdržálek. Fast mu-calculus model checking when tree-width is bounded. In W. A. Hunt and F. Somenzi, editors, *CAV 2003, Proceedings*, pages 80–92, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. doi:10.1007/978-3-540-45069-6_7.

23 S. Ronchi Della Rocca, editor. *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013. URL: http://www.dagstuhl.de/dagpub/978-3-939897-60-6.

24 H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logic*. In S. Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, volume 82 of *Studies in Logic and the Foundations of Mathematics*, pages 110–143. Elsevier, 1975. doi:10.1016/S0049-237X(08)70728-6.

25 L. Santocanale. μ-Bicomplete Categories and Parity Games. *ITA*, 36(2):195–227, 2002. Extended version appeared as LaBRI report RR-1281-02 is available from Hal: https://hal.archives-ouvertes.fr/hal-01376731. doi:10.1051/ita:2002010.

26 L. Santocanale. Completions of μ-algebras. *Ann. Pure Appl. Logic*, 154(1):27–50, 2008. doi:10.1016/j.apal.2007.11.001.

27 T. Studer. On the proof theory of the modal mu-calculus. *Studia Logica*, 89(3):343–363, 2008.

28 A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

29 S. K. Thomason. Independent propositional modal logics. *Studia Logica*, 39(2):143–144, 1980. doi:10.1007/BF00370317.

30 I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation*, 157(1):142–182, 2000. doi:10.1006/inco.1999.2836.

Taylor Expansion, β-Reduction and Normalization

Lionel Vaux

Aix Marseille Univ, CNRS, Centrale Marseille, I2M, Marseille, France

Abstract

We introduce a notion of reduction on resource vectors, *i.e.* infinite linear combinations of resource λ-terms. The latter form the multilinear fragment of the differential λ-calculus introduced by Ehrhard and Regnier, and resource vectors are the target of the Taylor expansion of λ-terms. We show that the reduction of resource vectors contains the image, through Taylor expansion, of β-reduction in the algebraic λ-calculus, *i.e.* λ-calculus extended with weighted sums: in particular, Taylor expansion and normalization commute. We moreover exhibit a class of algebraic λ-terms, having a normalizable Taylor expansion, subsuming both arbitrary pure λ-terms, and normalizable algebraic λ-terms. For these, we prove the commutation of Taylor expansion and normalization in a more denotational sense, mimicking the Böhm tree construction.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases lambda-calculus, non-determinism, normalization, denotational semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.39

1 Introduction

Quantitative semantics was first proposed by Girard [15] as an alternative to domains and continuous functionals, for defining denotational models of λ-calculi with a natural interpretation of non-determinism: a type is given by a collection of "atomic states"; a term of type A is then represented by a vector (*i.e.* a possibly infinite formal linear combination) of states. The main matter is the treatment of the function space: the construction requires the interpretation of function terms to be analytic, *i.e.* defined by power series. This interpretation of λ-terms was at the origin of linear logic: the model of coherence spaces was introduced as a simplified, qualitative version of quantitative semantics [14, Appendix C].

Dealing with power series, quantitative semantics must account for infinite sums. Girard's original model can be seen as a special case of Joyal's analytic functors [17]: in particular, coefficients are sets and infinite sums are given by coproducts. This allows to give an interpretation to fixed point operators and to the pure, untyped λ-calculus. On the other hand, it does not provide a natural way to deal with weighted (*e.g.*, probabilistic) non-determinism, where coefficients are taken in an external semiring of scalars.

In the early 2000's, Ehrhard introduced alternative presentations of quantitative semantics [8, 9], limited to a typed setting, but where types can be interpreted as vector spaces, or more generally semimodules over an arbitrary fixed semiring; these spaces are moreover equipped with a linear topology, allowing to interpret proofs as linear and continuous maps, in a standard sense. In this setting, the formal differentiation of power series recovers its usual meaning of linear approximation of a function, and the Taylor expansion formula holds: if ϕ is analytic then $\phi(\alpha) = \sum_{n\in\mathbf{N}} \frac{1}{n!}(\phi^{(n)}(0)) \cdot \alpha^n$ where $\phi^{(n)}(0)$ is the n-th derivative of ϕ computed at 0, which is an n-linear map, and α^n is the n-th tensor power of α.

Ehrhard and Regnier gave a computational meaning to such derivatives by introducing linearized variants of application and substitution in the λ-calculus, which led to the dif-

ferential λ-calculus [11], and then the resource λ-calculus [13] – the latter retains iterated derivatives at zero as the only form of application. They were then able to recast the above Taylor expansion formula in a syntactic, untyped setting: to every λ-term M, they associate a vector $\Theta(M)$ of resource λ-terms, *i.e.* terms of the resource λ-calculus.

The Taylor expansion of a λ-term can be seen as an intermediate, infinite object, between the term and its denotation in quantitative semantics. Indeed, resource terms still retain a dynamics, even if a very simple, finitary one: the size of terms is strictly decreasing under reduction. Furthermore, normal resource terms are in close relationship with the atomic states of quantitative semantics of the pure λ-calculus (or equivalently with the elements of a reflexive object in the relational model [3]; or with normal type derivations in a non-idempotent intersection type system [5]). So the normal form of $\Theta(M)$ should coincide with the denotation of M, which allows for a very generic description of quantitative semantics.

Other approaches to quantitative semantics impose a constraint *a priori*, either on the computational model or on the treatment of scalar coefficients: *e.g.*, the denotational model of finiteness spaces [9] is, by design, limited to strongly normalizing computation; probabilistic coherence spaces [4] can accommodate fixpoints but are limited to probabilistic distributions of terms; weighted relational models [20, 19] rely on a completeness property of the semiring; and Girard's original quantitative semantics [15] used sets as coefficients.

The "normalization of Taylor expansion" approach is more permissive. For instance, we can consider the algebraic λ-calculus [24], a very generic, non-uniform extension of λ-calculus, augmenting the syntax with formal finite linear combinations of terms: the Taylor expansion operator extends naturally to this setting. Of course, there is a price attached to such liberality: in general, the normal form of a vector of resource λ-terms is not well defined; and it is not difficult to find algebraic λ-terms whose Taylor expansion is not normalizable.

Ehrhard and Regnier proved, however, that the Taylor expansion of a pure λ-term M is always normalizable [13]. This can be seen as a new proof of the fact that Girard's quantitative semantics of pure λ-terms uses finite cardinals only [16]. They moreover established that this normal form is exactly the Taylor expansion of the Böhm tree $\mathsf{BT}(M)$ of M [12]. Both results rely heavily on the uniformity of the pure λ-calculus: all resource terms in $\Theta(M)$ follow a single tree pattern. This is a bit disappointing since quantitative semantics was introduced as a model of non-determinism, which is incompatible with uniformity.

In the present paper, we transpose Ehrhard and Regnier's results to the full algebraic λ-calculus, which requires the development of a new proof technique, not relying on uniformity:

1. we prove that normalizable algebraic λ-terms have a normalizable Taylor expansion (Theorem 47, Section 7);
2. we introduce a notion of reduction on vectors of resource λ-terms (Section 5), containing the image of β-reduction on algebraic λ-terms by Taylor expansion (Section 6);
3. we deduce that, for normalizable algebraic λ-terms, normalization and Taylor expansion commute on the nose (Section 7);
4. we introduce the class of *hereditarily determinable* terms, containing in particular all pure λ-terms as well as all normalizable algebraic λ-terms, together with a notion of finite normal form approximants, called *d-determinate forms*; and we prove that if M is hereditarily determinable, then $\Theta(M)$ is normalizable, and its normal form is entirely characterized by the d-determinate forms of M (Section 8).

Step 1 is a generalization of previous work by Ehrhard [10] in a typed setting, and of our own work with Pagani and Tasson [22] for strongly normalizable terms. Let us stress that, in those previous contributions, the actual commutation of normalization and Taylor expansion

was never considered, let alone any analogue of step 4. The results of Ehrhard and Regnier about the pure λ-calculus [13, 12] provided analogues of steps 1 and 4 (which entails 3), using uniformity to skip step 2. When restricted to pure λ-terms, our results actually provide a new proof, using very different tools, that the normal form of $\Theta(M)$ is isomorphic to $\mathsf{BT}(M)$.

In their full extent, our results provide a generalization of the notion of non-deterministic Böhm tree [6] in a weighted, quantitative setting. This may appeal to anyone with an interest in denotational semantics or infinitary normal forms, in a non-deterministic setting, but the techniques we rely on may be familiar only to a much narrower audience, so we start the paper by reviewing them briefly: in Section 2 we recall the notion of finiteness space and detail the case of linear-continuous maps defined by summable families of vectors, which is pervasive in the paper; we review the syntax and dynamics of the resource λ-calculus in Section 3; then we present the Taylor expansion construction and some of its basic properties in Section 4 – this includes results that might be considered folklore by some, but for which we found no previously published reference.

2 Finiteness spaces and summable functions

Finiteness spaces were introduced by Ehrhard [9] as a model of linear logic, giving an account of quantitative semantics in a standard linear algebraic setting. In short, types are interpreted as topological vector spaces of a particular form, linear logic proofs are interpreted as linear-continuous maps, and λ-terms as analytic maps, subject to Taylor expansion.

For the rest of the paper, $\mathbf{S}$ will denote an arbitrary commutative semiring $\mathbf{S} = (\underline{\mathbf{S}}, 0_{\mathbf{S}}, +_{\mathbf{S}}, 1_{\mathbf{S}}, \cdot_{\mathbf{S}})$. We will often abuse notation and identify $\mathbf{S}$ with its carrier set $\underline{\mathbf{S}}$; we may also omit the subscripts on $+$, $\cdot$, 0 and 1, and write ab for $a \cdot b$. Given a set X, we write $\mathbf{S}^X$ for the semimodule of formal linear combinations of elements in X: a *vector* $\xi \in \mathbf{S}^X$ is nothing but an X-indexed family of scalars $(\xi_x)_{x\in X}$, that we may also denote by $\sum_{x\in X} \xi_x.x$. The *support* $|\xi|$ of a vector $\xi \in \mathbf{S}^X$ is the set of elements of X having a non-zero coefficient in ξ: $|\xi| := \{x \in X \,;\, \xi_x \neq 0\}$. We write $\mathbf{S}[X]$ for the set of vectors with finite support.

If X is a set, we write $\mathfrak{P}(X)$ for the powerset of X. We call *structure* on X any $\mathfrak{X} \subseteq \mathfrak{P}(X)$ and then we write $\mathfrak{X}^\perp := \{x' \subseteq X \,;\ \text{for all } x \in \mathfrak{X},\ x \cap x' \text{ is finite}\}$. A *relational finiteness space* $\mathcal{X}$ is a pair $(\|\mathcal{X}\|, \mathfrak{F}(\mathcal{X}))$, where $\|\mathcal{X}\|$ is a set (the *web* of $\mathcal{X}$) and $\mathfrak{F}(\mathcal{X})$ is a structure on $\|\mathcal{X}\|$ such that $\mathfrak{F}(\mathcal{X}) = \mathfrak{F}(\mathcal{X})^{\perp\perp}$: $\mathfrak{F}(\mathcal{X})$ is called the *finiteness structure* of $\mathcal{X}$, and we say $x \subseteq \|\mathcal{X}\|$ is *finitary* in $\mathcal{X}$ iff $x \in \mathfrak{F}(\mathcal{X})$. The *finiteness space* generated by $\mathcal{X}$ is the set $\mathbf{S}\langle\mathcal{X}\rangle$ of vectors on $\|\mathcal{X}\|$ with finitary support: $\xi \in \mathbf{S}\langle\mathcal{X}\rangle$ iff $|\xi| \in \mathfrak{F}(\mathcal{X})$. Finitary subsets are downwards closed for inclusion, and finite unions of finitary subsets are finitary, hence $\mathbf{S}\langle\mathcal{X}\rangle$ is a subsemimodule of $\mathbf{S}^X$. The least (resp. greatest) finiteness structure on X is the set $\mathfrak{P}_f(X)$ of finite subsets of X (resp. $\mathfrak{P}(X)$), generating the finiteness space $\mathbf{S}[X]$ (resp. $\mathbf{S}^X$).

We do not describe the whole category of finiteness spaces and linear-continuous maps here. In particular we do not recall the details of the linear topology induced on $\mathbf{S}\langle\mathcal{X}\rangle$ by $\mathfrak{F}(\mathcal{X})$: the reader may refer to Ehrhard's original paper. In the remaining of this section, we focus on a very particular case, where the finiteness structure on base types is trivial: linear-continuous maps are then univocally generated by summable functions.

▶ **Definition 1.** Let $\vec{\xi} = (\xi_i)_{i\in I} \in (\mathbf{S}^X)^I$ be a family of vectors: write $\xi_i = \sum_{x\in X} \xi_{i,x}.x$. We say $\vec{\xi}$ is *summable* if, for all $x \in X$, $\{i \in I \,;\, x \in |\xi_i|\}$ is finite. In this case, we define the sum $\sum \vec{\xi} = \sum_{i\in I} \xi_i \in \mathbf{S}^X$ in the obvious, pointwise way: $(\sum \vec{\xi})_x := \sum_{i\in I} \xi_{i,x}$.

Of course, any finite family of vectors is summable and we use standard notations for finite sums. The reader can moreover check that the family $\vec{\xi}$ is summable iff the set $\{(i, x) \in$

$I \times X \,;\, \xi_{i,x} \neq 0\}$ is finitary in the relational arrow finiteness space $(I, \mathfrak{P}(I)) \multimap (X, \mathfrak{P}(X))$ as defined by Ehrhard [9, in particular Lemma 3]. In particular, summable families form a finiteness space, hence an $\mathbf{S}$-semimodule. If $f : X \to \mathbf{S}^Y$ is a summable function (*i.e.* the family $(f(x))_{x\in X}$ is summable) and $\xi \in \mathbf{S}^X$, we write $f \cdot \xi := \sum_{x\in X} \xi_x.f(x)$.

▶ **Definition 2.** Let $\phi : \mathbf{S}^X \to \mathbf{S}^Y$. We say ϕ is *linear-continuous* if, for all summable family $(\xi_i)_{i\in I} \in (\mathbf{S}^X)^I$, the family $(\phi(\xi_i))_{i\in I}$ is summable and, for all family of scalars, $(a_i)_{i\in I} \in \mathbf{S}^I$, we have $\phi(\sum_{i\in I} a_i.\xi_i) = \sum_{i\in I} a_i.\phi(\xi_i)$.

One might check that a map $\phi : \mathbf{S}^X \to \mathbf{S}^Y$ is linear-continuous in the sense of Definition 2 iff it is linear and continuous in the sense of the linear topology of finiteness spaces, observing that the topology on $\mathbf{S}^X = \mathbf{S}\langle(X, \mathfrak{P}(X))\rangle$ is the product topology [9, Section 3]. As a general fact in finiteness spaces, linear-continuous maps are those defined by finitary matrices:

▶ **Lemma 3.** *If $\phi : \mathbf{S}^X \to \mathbf{S}^Y$ is linear-continuous then its restriction $\phi|_X : X \to \mathbf{S}^Y$ is summable and $\phi(\xi) = \phi|_X \cdot \xi$. Conversely, if $f : X \to \mathbf{S}^Y$ is summable then $\xi \mapsto f \cdot \xi$ is linear-continuous.*

It is easy to generalize Definitions 1 and 2 to n-ary functions. Following the constructions of finiteness spaces, n-ary summable functions $f : X_1 \times \cdots \times X_n \to \mathbf{S}^Y$ correspond with n-linear-hypocontinuous maps [9, Section 3] from $\mathbf{S}^{X_1} \times \cdots \times \mathbf{S}^{X_n}$ to $\mathbf{S}^Y$, rather than the more restrictive multilinear-continuous maps. In the simpler setting of summable functions, though, both notions coincide, since $\mathbf{S}^X$ is always locally linearly compact [9, Proposition 15], so the previous lemma still holds for n-ary functions. In the remaining, we will thus identify n-ary summable functions with n-linear-continuous maps.

3 The resource λ-calculus

We fix an infinite, denumerable set $\mathcal{V}$ of variables, that we denote by small letters x, y, z. We define the sets Δ of *resource terms* and $!\Delta$ of *resource monomials* simultaneously as follows:[1]

$$\Delta \ni s, t, u, v, w ::= x \mid \lambda x\, s \mid \langle s\rangle\ \overline{t} \qquad \text{and} \qquad !\Delta \ni \overline{s}, \overline{t}, \overline{u}, \overline{v}, \overline{w} ::= [] \mid [s] \cdot \overline{t}.$$

Terms are considered up to α-equivalence and monomials up to permutativity: we write $[t_1, \dots, t_n]$ for $[t_1]\cdot(\cdots([t_n]\cdot[]))$ and equate $[t_1, \dots, t_n]$ with $[t_{f(1)}, \dots, t_{f(n)}]$ for all permutation f of $\{1, \dots, n\}$, so that resource monomials coincide with finite multisets of resource terms.[2] We will then write $\overline{s} \cdot \overline{t}$ for the multiset union of $\overline{s}$ and $\overline{t}$, and $\#[s_1, \dots, s_n] := n$.

We call resource expression any resource term or resource monomial and write $(!)\Delta$ for either Δ or $!\Delta$: whenever we use this notation several times in the same context, all occurrences consistently denote the same set.

▶ **Definition 4.** We define by induction over a resource expression $e \in (!)\Delta$, its *size* $\mathbf{s}(e) \in \mathbf{N}$ and its *height* $\mathbf{h}(e) \in \mathbf{N}$, by setting $\mathbf{s}(x) := 1$ and $\mathbf{h}(x) := 1$ and, supposing $\overline{t} = [t_1, \dots, t_n]$:

$$\begin{array}{lll} \mathbf{s}(\lambda x\, s) := 1 + \mathbf{s}(s) & \mathbf{s}(\langle s\rangle\ \overline{t}) := 1 + \mathbf{s}(s) + \mathbf{s}(\overline{t}) & \mathbf{s}(\overline{t}) := \mathbf{s}(t_1) + \cdots + \mathbf{s}(t_n) \\ \mathbf{h}(\lambda x\, s) := 1 + \mathbf{h}(s) & \mathbf{h}(\langle s\rangle\ \overline{t}) := \max(\mathbf{h}(s), 1 + \mathbf{h}(\overline{t})) & \mathbf{h}(\overline{t}) := \max(\mathbf{h}(t_1), \dots, \mathbf{h}(t_n)). \end{array}$$

[1] We use a variant of BNF notation: *e.g.*, $!\Delta$ is inductively generated by $[]$ and addition of a resource term to a monomial; and we denote monomials by overlined letters $\overline{s}, \overline{t}, \overline{u}, \overline{v}, \overline{w}$, possibly with sub- and superscripts.

[2] Resource monomials are also called *bags*, *bunches* or *poly-terms* in the literature, but we prefer to strengthen the analogy with power series here.

$$\frac{}{\langle \lambda x\, s\rangle\ \overline{t} \to_\partial \partial_x s \cdot \overline{t}} \qquad \frac{s \to_\partial \sigma'}{\lambda x\, s \to_\partial \lambda x\, \sigma'} \qquad \frac{s \to_\partial \sigma'}{\langle s\rangle\ \overline{t} \to_\partial \langle \sigma'\rangle\ \overline{t}} \qquad \frac{\overline{t} \to_\partial \overline{\tau}'}{\langle s\rangle\ \overline{t} \to_\partial \langle s\rangle\ \overline{\tau}'} \qquad \frac{s \to_\partial \sigma'}{[s]\cdot\overline{t} \to_\partial [\sigma']\cdot\overline{t}}$$

Figure 1 Rules of resource reduction $\to_\partial$.

In particular $\mathbf{s}(s) > 0$ and $\mathbf{h}(s) > 0$ for all $s \in \Delta$, and $\mathbf{s}(\overline{s}) \geq \#\overline{s}$ for all $\overline{s} \in !\Delta$. For all $e \in (!)\Delta$, we write $\mathsf{fv}(e)$ for the set of its free variables and $\mathbf{n}_x(e) \in \mathbf{N}$ for the number of free occurrences of x in e. It should be clear that $\mathbf{n}_x(e) \leq \mathbf{s}(e)$, and $x \in \mathsf{fv}(e)$ iff $\mathbf{n}_x(e) \neq 0$.

In the resource λ-calculus, the substitution $e[s/x]$ of a term s for a variable x in e admits a linear counterpart: this operator was initially introduced in the differential λ-calculus [11] in the form of a partial differentiation operation, reflecting the interpretation of λ-terms in quantitative semantics. Here we only consider those derivatives involved in the Taylor expansion of λ-terms: iterated derivatives at 0, *e.g.* $(\frac{\partial^n e}{\partial x^n}\cdot(u_1,\dots,u_n))[0/x]$, that are n-linear symmetric in the u_i's, and that we denote more concisely by $\partial_x e \cdot [u_1,\dots,u_n]$.

To introduce this construct, we need to consider formal finite sums of resource expressions, and to extend all syntactic constructs by linearity: if $\sigma = \sum_{i=1}^n s_i \in \mathbf{N}[\Delta]$ and $\overline{\tau} = \sum_{j=1}^p \overline{t}_j \in \mathbf{N}[!\Delta]$, we set $\lambda x\, \sigma := \sum_{i=1}^n \lambda x\, s_i$, $\langle\sigma\rangle\ \overline{\tau} := \sum_{i=1}^n \sum_{j=1}^p \langle s_i\rangle\ \overline{t}_j$ and $[\sigma]\cdot\overline{\tau} := \sum_{i=1}^n \sum_{j=1}^p [s_i]\cdot\overline{t}_j$.

▶ **Definition 5.** If $\overline{u} = [u_1,\dots,u_n] \in !\Delta$ and $1 \leq i_1 < \dots < i_l \leq n$, we write $\overline{u}_{\{i_1,\dots,i_l\}} := [u_{i_1},\dots,u_{i_l}]$. Then we define the *$n$-linear substitution* of $\overline{u}$ for x in e inductively as follows: [3]

$$\partial_x y \cdot \overline{u} := \begin{cases} y & \text{if } y \neq x \text{ and } n = 0 \\ u_1 & \text{if } y = x \text{ and } n = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \partial_x \langle s\rangle\ \overline{t} \cdot \overline{u} := \sum_{(I_0,I_1) \text{ partition of } \{1,\dots,n\}} \langle \partial_x s \cdot \overline{u}_{I_0}\rangle\ \partial_x \overline{t} \cdot \overline{u}_{I_1}$$

$$\partial_x \lambda y\, s \cdot \overline{u} := \lambda y\, (\partial_x s \cdot \overline{u}) \qquad \partial_x [s_1,\dots,s_k] \cdot \overline{u} := \sum_{(I_1,\dots,I_k) \text{ partition of } \{1,\dots,n\}} [\partial_x s_1 \cdot \overline{u}_{I_1},\dots,\partial_x s_k \cdot \overline{u}_{I_k}].$$

▶ **Lemma 6.** *If $e' \in |\partial_x e \cdot \overline{u}|$ then $\mathbf{n}_x(e) = \#\overline{u}$ and $\mathsf{fv}(e') = (\mathsf{fv}(e) \setminus \{x\}) \cup \mathsf{fv}(\overline{u})$. Moreover, we have $\mathbf{s}(e') = \mathbf{s}(e) + \mathbf{s}(\overline{u}) - \#\overline{u}$ and $\mathbf{h}(e) \leq \mathbf{h}(e') \leq \mathbf{h}(e) + \mathbf{h}(\overline{u}) - 1$.*

If $\to$ is a reduction relation, we write $\to^?$ (resp. $\to^*$) for its reflexive (resp. reflexive and transitive) closure. A redex is a term of the form $\langle \lambda x\, t\rangle\ \overline{u}$, and its reduct is $\partial_x t \cdot \overline{u} \in \mathbf{N}[\Delta]$.

▶ **Definition 7.** We define the *resource reduction* relation $\to_\partial \subseteq (!)\Delta \times \mathbf{N}[(!)\Delta]$ inductively by the rules of Fig. 1. We then extend $\to_\partial$ to finite sums, setting $\epsilon \to_\partial \epsilon'$ if $\epsilon = \sum_{i=0}^n e_i$ and $\epsilon' = \sum_{i=0}^n \epsilon'_i$ with $e_0 \to_\partial \epsilon'_0$ and, for all $i \geq 1$, $e_i \to_\partial^? \epsilon'_i$.

The way the reduction is extended to sums is sufficient [13] to ensure the strong confluence of $\to_\partial^?$. Moreover, the effect of reduction on the size of terms is very regular:

▶ **Lemma 8.** *If $e \to_\partial \epsilon'$ and $e' \in |\epsilon'|$ then $\mathsf{fv}(e') = \mathsf{fv}(e)$ and $\mathbf{s}(e') + 2 \leq \mathbf{s}(e) \leq 2\mathbf{s}(e') + 2$.*

We will write $e \geq_\partial e'$ if $e \to_\partial^* \epsilon'$ with $e' \in |\epsilon'|$. We obtain that $\{e'\ ;\ e \geq_\partial e'\}$ is finite hence $\to_\partial$ is strongly normalizing: we write $\mathsf{NF}(\epsilon)$ for the normal form of ϵ.

4 Resource vectors and Taylor expansion of algebraic λ-terms

In order to write down the Taylor expansion formula, we will need inverses of natural numbers to be available: say $\mathbf{S}$ *has fractions* if there is a semiring morphism from $\mathbf{Q}^+$ (the semiring

[3] We adhere to Barendregt's convention and, *e.g.*, implicitly assume $y \notin \mathsf{fv}(\overline{u})$ in the abstraction case.

of non negative rational numbers) to $\mathbf{S}$. We abuse notation and identify rationals with their images; moreover, we silently assume $\mathbf{S}$ has fractions whenever we use them.

4.1 Resource vectors

A vector $\sigma = \sum_{s\in\Delta} \sigma_s.s$ of resource terms will be called a *term vector*. Similarly, we will call *monomial vector* any vector of resource monomials. A *resource vector* will be any of a term vector or a monomial vector and, again, we will write $\mathbf{S}^{(!)\Delta}$ for either $\mathbf{S}^{\Delta}$ or $\mathbf{S}^{!\Delta}$. The syntactic constructs are extended to resource vectors by linearity: for all $\sigma \in \mathbf{S}^{\Delta}$ and $\overline{\sigma}, \overline{\tau} \in \mathbf{S}^{!\Delta}$, we set $\lambda x\, \sigma := \sum_{s\in\Delta} \sigma_s.\lambda x\, s$, $\langle\sigma\rangle\ \overline{\tau} := \sum_{s\in\Delta,\overline{t}\in!\Delta} \sigma_s \overline{\tau}_{\overline{t}}.\ \langle s\rangle\ \overline{t}$, $[\sigma] := \sum_{s\in\Delta} \sigma_s.[s]$ and $\overline{\sigma}\cdot\overline{\tau} := \sum_{\overline{s},\overline{t}\in!\Delta} \overline{\sigma}_{\overline{s}}\overline{\tau}_{\overline{t}}.\overline{s}\cdot\overline{t}$. For these sums to be meaningful, we need to prove that the constructors of the calculus define summable functions, which is quite straightforward: *e.g.*, for every $\overline{u} \in !\Delta$ there is at most one s with $\overline{u} \in |[s]|$ (*i.e.* $\overline{u} = [s]$); and there are finitely many pairs $(\overline{s},\overline{t})$ with $\overline{u} \in |\overline{s}\cdot\overline{t}|$ (*i.e.* $\overline{u} = \overline{s}\cdot\overline{t}$). Similarly, for linear substitution:

▶ **Lemma 9.** *The family $(\partial_x e\cdot\overline{s})_{(e,\overline{s})\in(!)\Delta\times!\Delta}$ is summable.*

Proof. First observe that we can consider $\partial_x e\cdot\overline{s} \in \mathbf{N}[(!)\Delta] \subseteq \mathbf{S}^{(!)\Delta}$ via the unique semiring morphism from $\mathbf{N}$ into $\mathbf{S}$. By Lemma 6, if $e' \in |\partial_x e\cdot\overline{s}|$ then $\mathsf{fv}(e) \subseteq \mathsf{fv}(e')\cup\{x\}$, $\mathsf{fv}(\overline{s}) \subseteq \mathsf{fv}(e')$, $\mathbf{s}(e) \leq \mathbf{s}(e')$ and $\mathbf{s}(\overline{s}) \leq \mathbf{s}(e')$: e' being fixed, there are finitely many such pairs $(e,\overline{s})$. ◀

We thus obtain a bilinear-continuous map: $\partial_x\epsilon\cdot\overline{\sigma} := \sum_{e\in(!)\Delta,\overline{s}\in!\Delta} \epsilon_e\overline{\sigma}_{\overline{s}}.\partial_x e\cdot\overline{s}$. By contrast, the usual substitution is not linear, so it must be defined directly for resource vectors.

▶ **Definition 10.** We define the substitution $e[\sigma/x] \in \mathbf{S}^{(!)\Delta}$ of $\sigma \in \mathbf{S}^{\Delta}$ for a variable x in $e \in (!)\Delta$ inductively as follows (assuming $x \neq y$ and $\overline{t} = [t_1,\dots,t_n]$):

$$x[\sigma/x] := \sigma \qquad (\lambda y\, s)[\sigma/x] := \lambda y\, s[\sigma/x]$$
$$y[\sigma/x] := y \qquad (\langle s\rangle\ \overline{t})[\sigma/x] := \langle s[\sigma/x]\rangle\ \overline{t}[\sigma/x] \qquad \overline{t}[\sigma/x] := [t_1[\sigma/x],\dots,t_n[\sigma/x]].$$

▶ **Lemma 11.** *For all $e \in (!)\Delta$, $x \in \mathcal{V}$ and $\sigma \in \mathbf{S}^{\Delta}$: if $x \notin \mathsf{fv}(e)$ then $e[\sigma/x] = e$; otherwise, if $x \in \mathsf{fv}(e)$ then $e[0/x] = 0$. Moreover, for all $e' \in |e[\sigma/x]|$, $\mathsf{fv}(e) \subseteq \mathsf{fv}(e')\cup\{x\}$, $\mathsf{fv}(e') \subseteq \mathsf{fv}(e)\cup\mathsf{fv}(\sigma)$ and $\mathbf{s}(e') \geq \mathbf{s}(e)$.*

A consequence of the previous lemma is that the function $e \in (!)\Delta \mapsto e[\sigma/x] \in \mathbf{S}^{(!)\Delta}$ is summable: we thus write $\epsilon[\sigma/x] := \sum_{e\in\mathbf{S}^{(!)\Delta}} \epsilon_e.e[\sigma/x]$.

4.2 Promotion

For all $\sigma \in \mathbf{S}^{\Delta}$, we define $\sigma^n \in \mathbf{S}^{!\Delta}$ by induction on $n \in \mathbf{N}$: $\sigma^0 = []$ and $\sigma^{n+1} = [\sigma]\cdot\sigma^n$. Observe that the family $(\sigma^n)_{n\in\mathbf{N}}$ of monomial vectors is summable because the supports $|\sigma^n|$ for $n \in \mathbf{N}$ are pairwise disjoint. We then define the promotion of σ as $\sigma^! := \sum_{n\in\mathbf{N}} \frac{1}{n!}\sigma^n$.

▶ **Lemma 12.** *For all σ and $\tau \in \mathbf{S}^{\Delta}$, $\sigma^![\tau/x] = (\sigma[\tau/x])^!$.*

Proof. By the linear-continuity of $\epsilon \mapsto \epsilon[\sigma/x]$, it is sufficient to prove that $\sigma^n[\tau/x] = (\sigma[\tau/x])^n$ which follows from the n-linear-continuity of $(\sigma_1,\dots,\sigma_n) \mapsto [\sigma_1,\dots,\sigma_n]$. ◀

For all $k, l_1, \dots, l_n \in \mathbf{N}$ such that $k = l_1 + \cdots + l_n$, we write $\left[\begin{smallmatrix} k \\ l_1,\dots,l_n \end{smallmatrix}\right] := \frac{k!}{\prod_{i=1}^{n} l_i!}$, which is always an integer. These *multinomial coefficients* generalize the binomial coefficients: $\binom{k}{l} = \left[\begin{smallmatrix} k \\ l,k-l \end{smallmatrix}\right]$. They arise naturally in the description of various quantitative denotational semantics of linear logic and λ-calculus [9, 4], more specifically in the interpretation of the promotion rule. Here we will use the following result, which boils down to an exercise on the combinatorics of the partitions of $\{1,\dots,k\}$:

▶ **Lemma 13.** *If $s \in \Delta$, $\bar{t} = [t_1, \dots, t_n] \in !\Delta$, and $\rho \in \mathbf{S}^{\Delta}$, then*

$$\partial_x \langle s \rangle\ \bar{t}\cdot\rho^k = \sum_{l=0}^{k} [\,{}^{k}_{l,k-l}\,] \left\langle \partial_x s \cdot \rho^l \right\rangle\ \partial_x \bar{t}\cdot\rho^{k-l} \text{ and } \partial_x \bar{t}\cdot\rho^k = \sum_{\substack{l_1,\dots,l_n \in \mathbf{N} \\ l_1+\dots+l_n=k}} [\,{}^{k}_{l_1,\dots,l_n}\,][\partial_x t_1\cdot\rho^{l_1}, \dots, \partial_x t_n\cdot\rho^{l_n}].$$

▶ **Lemma 14.** *The following identities hold (assuming $x \neq y$ and $\overline{\tau} = [\tau_1, \dots, \tau_n]$):*

$$\partial_x x \cdot \rho^! = \rho \qquad \partial_x \lambda y\, \sigma \cdot \rho^! = \lambda y\ \left(\partial_x \sigma \cdot \rho^!\right) \qquad \partial_x \overline{\tau} \cdot \rho^! = [\partial_x \tau_1 \cdot \rho^!, \dots, \partial_x \tau_n \cdot \rho^!].$$
$$\partial_x y \cdot \rho^! = y \qquad \partial_x \langle \sigma \rangle\ \overline{\tau} \cdot \rho^! = \left\langle \partial_x \sigma \cdot \rho^! \right\rangle\ \partial_x \overline{\tau} \cdot \rho^!$$

Proof. Since each syntactic constructor is multilinear-continuous and $\epsilon \mapsto \partial_x \epsilon \cdot \rho^!$ is linear-continuous for a fixed ρ, it is sufficient to consider the case of $\partial_x e \cdot \rho^!$ for a resource expression $e \in (!)\Delta$. First observe that, by Lemma 6, if $k = \mathbf{n}_x(e)$ then $\partial_x e \cdot \rho^! = \frac{1}{k!}.\partial_x e \cdot \rho^k$. In particular the case of variables is straightforward.

In the abstraction case, $\partial_x \lambda x\, s \cdot \rho^! = \lambda x\ \left(\partial_x s \cdot \rho^!\right)$ follows from the linear-continuity of multilinear substitution and the fact that $\partial_x \lambda x\, s \cdot \bar{t} = \lambda x\ \left(\partial_x s \cdot \bar{t}\right)$ for all $\bar{t} \in !\Delta$.

If $e = \langle s \rangle\ \bar{t}$, write $l = \mathbf{n}_x(s)$ and $m = \mathbf{n}_x(\bar{t})$. Then $k = l+m$ and by the previous lemma and Lemma 6 again $\partial_x e \cdot \rho^k = [\,{}^{k}_{l,m}\,].\left\langle \partial_x s \cdot \rho^l \right\rangle\ \partial_x \bar{t} \cdot \rho^m$ and then $\frac{1}{k!}.\partial_x e \cdot \rho^k = \left\langle \frac{1}{l!}.\partial_x s \cdot \rho^l \right\rangle\ \frac{1}{m!}\partial_x \bar{t} \cdot \rho^m$. The case of monomials is similar. ◀

▶ **Lemma 15.** *For all $\epsilon \in \mathbf{S}^{(!)\Delta}$ and $\sigma \in \mathbf{S}^{\Delta}$, $\epsilon[\sigma/x] = \partial_x \epsilon \cdot \sigma^!$.*

Proof. By the linear-continuity of $\epsilon \mapsto \partial_x \epsilon \cdot \sigma^!$ and $\epsilon \mapsto \epsilon[\sigma/x]$, it suffices to show $e[\sigma/x] = \partial_x e \cdot \sigma^!$ for $e \in (!)\Delta$: the proof is then by induction on e, using Lemma 14 in each case. ◀

By Lemma 12, we thus obtain $\partial_x \sigma^! \cdot \tau^! = \left(\partial_x \sigma \cdot \tau^!\right)^!$ which can be seen as a counterpart of the functoriality of promotion in linear logic. To our knowledge it is the first published proof of such a result for resource vectors. This will enable us to prove the commutation of Taylor expansion and substitution (Lemma 17), another unsurprising yet non-trivial result.

4.3 Taylor expansion of algebraic λ-terms

Since resource vectors form a module, there is no reason to restrict the source language of Taylor expansion to the pure λ-calculus. We will thus consider the following set of terms:

$$\Sigma_{\mathbf{S}} \ni M, N, P ::= x \mid \lambda x\, M \mid (M)\ N \mid 0 \mid a.M \mid M + N$$

where a ranges in $\mathbf{S}$. For now, terms are considered up to the usual α-equivalence only: the null term 0, scalar multiplication $a.M$ and sum $M + N$ are purely syntactic constructs.

▶ **Definition 16.** We define the Taylor expansion $\Theta(M) \in \mathbf{S}^{(!)\Delta}$ of $M \in \Sigma_{\mathbf{S}}$ by induction:

$$\Theta(x) := x \qquad \Theta(\lambda x\, M) := \lambda x\, \Theta(M) \qquad \Theta((M)\ N) := \langle \Theta(M) \rangle\ \Theta(N)^!$$
$$\Theta(0) := 0 \qquad \Theta(a.M) := a.\Theta(M) \qquad \Theta(M + N) := \Theta(M) + \Theta(N).$$

▶ **Lemma 17.** *For all $M, N \in \Sigma_{\mathbf{S}}$, $\Theta(M[N/x]) = \partial_x \Theta(M) \cdot \Theta(N)^! = \Theta(M)[\Theta(N)/x]$.*

Proof. By induction on M, using Lemmas 14 and 15. ◀

Write $M \simeq_\Theta N$ if $\Theta(M) = \Theta(N)$. This defines a *congruence* on terms: $\simeq_\Theta$ is an equivalence relation and $M \simeq_\Theta M'$ implies $\lambda x\, M \simeq_\Theta \lambda x\, M'$, $(M)\ N \simeq_\Theta (M')\ N$, $(N)\ M \simeq_\Theta (N)\ M'$, $a.M \simeq_\Theta a.M'$, $M + N \simeq_\Theta M' + N$ and $N + M \simeq_\Theta N + M'$. Write $\simeq_{\mathrm{v}}$ for the least

$$
\begin{array}{lccc}
\text{(a)} & 0 + M \simeq_{\mathrm{v}} M & M + N \simeq_{\mathrm{v}} N + M & (M + N) + P \simeq_{\mathrm{v}} M + (N + P) \\
& 0.M \simeq_{\mathrm{v}} 0 & 1.M \simeq_{\mathrm{v}} M & a.M + b.M \simeq_{\mathrm{v}} (a + b).M \\
& a.0 \simeq_{\mathrm{v}} 0 & a.(b.M) \simeq_{\mathrm{v}} ab.M & a.(M + N) \simeq_{\mathrm{v}} a.M + a.N \\
\text{(b)} & \lambda x\, 0 \simeq_{\mathrm{v}} 0 & \lambda x\ (a.M) \simeq_{\mathrm{v}} a.\lambda x\, M & \lambda x\ (M + N) \simeq_{\mathrm{v}} \lambda x\, M + \lambda x\, N \\
& (0)\ P \simeq_{\mathrm{v}} 0 & (a.M)\ P \simeq_{\mathrm{v}} a.\,(M)\ P & (M + N)\ P \simeq_{\mathrm{v}} (M)\ P + (N)\ P
\end{array}
$$

Figure 2 Equations of vector λ-terms: (a) equations of $\mathbf{S}$-module; (b) linearity.

congruence containing the equations of Fig. 2, so that $M \simeq_{\mathrm{v}} N$ implies $M \simeq_{\Theta} N$. We call *vector λ-terms* the elements of the quotient $\Sigma_{\mathbf{S}}/\simeq_{\mathrm{v}}$, which forms an $\mathbf{S}$-semimodule: those are the terms of the previously studied algebraic λ-calculus [24, 1] (where they were called algebraic λ-terms; here we reserve this name for another, simpler notion).

Notice that Taylor expansion is not injective on vector λ-terms in general. For instance, if $\mathbf{S} = \mathbf{B}$ (the commutative semiring of booleans: $\underline{\mathbf{B}} = \{0, 1\}$, $+_{\mathbf{B}} = \vee$ and $\cdot_{\mathbf{B}} = \wedge$), we obtain $(x)\ 0 + (x)\ x \simeq_{\Theta} (x)\ x$. It is moreover well known [24, 2] that β-reduction in a semimodule of terms is inconsistent in presence of negative coefficients.

▶ **Example 18.** Consider $\delta_M := \lambda x\ (M + (x)\ x)$ and $\infty_M := (\delta_M)\ \delta_M$. Observe that ∞_M β-reduces to $M + \infty_M$. Then any congruence $\simeq$ on $\Sigma_{\mathbf{Z}}$ containing β-reduction and the equations of $\mathbf{Z}$-semimodule is inconsistent: $0 \simeq \infty_M - \infty_M \simeq M + \infty_M - \infty_M \simeq M$.

It is not our purpose here to survey the various possible approaches to the rewriting theory of β-reduction on vector λ-terms: we refer the reader to the literature on algebraic λ-calculi [24, 2, 1, 7] for several proposals. Our focus being on Taylor expansion, we rather propose to consider vector λ-terms as intermediate objects: the reduction relation induced on resource vectors by β-reduction through Taylor expansion should also contain β-reduction on vector terms – which is useful to understand what may go wrong. We still need to introduce some form of quotient in the syntax, though: otherwise, *e.g.*, $(\lambda x\, M + \lambda x\, N)\ P$ is normal.

We call *algebraic λ-terms* the elements of $\Lambda_{\mathbf{S}} := \Sigma_{\mathbf{S}}/\simeq_{+}$, where $\simeq_{+}$ denotes the least congruence containing the linearity equations of Fig. 2, group (b). We define the sets $\Sigma^{\mathrm{c}}_{\mathbf{S}}$ of *canonical terms* and $\Sigma^{\mathrm{s}}_{\mathbf{S}}$ of *simple canonical terms* simultaneously as follows:

$$\Sigma^{\mathrm{s}}_{\mathbf{S}} \ni S, T ::= x \mid \lambda x\, S \mid (S)\ M \qquad \text{and} \qquad \Sigma^{\mathrm{c}}_{\mathbf{S}} \ni M, N, P ::= S \mid 0 \mid a.M \mid M + N.$$

so that each algebraic term M admits a unique canonical $\simeq_{+}$-representative. Moreover, each simple canonical term $S \in \Sigma^{\mathrm{s}}_{\mathbf{S}}$ is of one of the following two forms:

- either $S = \lambda x_1 \cdots \lambda x_n\ (x)\ M_1 \cdots M_k$: S is a *head normal form*;
- or $S = \lambda x_1 \cdots \lambda x_n\ (\lambda x\, T)\ M_0 \cdots M_k$: $(\lambda x\, T)\ M_0$ is the *head redex* of S.

In the remaining of this paper we will systematically identify algebraic terms with their canonical representatives and keep $\simeq_{+}$ implicit. Moreover, we write $\Lambda^{\mathrm{s}}_{\mathbf{S}}$ for the set of simple algebraic λ-terms, *i.e.* those that admit a simple canonical representative.

By contrast with the above treatment of Taylor expansion, one can forget everything about coefficients and focus on sets of resource terms (identifying $\mathfrak{P}(\Delta)$ with $\mathbf{B}^{\Delta}$, we write, *e.g.*, $\lambda x\, \mathcal{S} = \{\lambda x\, s\ ;\ s \in \mathcal{S}\}$ if $\mathcal{S} \subseteq \Delta$):

▶ **Definition 19.** The Taylor support $\mathcal{T}(M) \subseteq \Delta$ of $M \in \Sigma_{\mathbf{S}}$ is defined inductively as follows:

$$
\begin{array}{lll}
\mathcal{T}(x) := \{x\} & \mathcal{T}(\lambda x\, M) := \lambda x\, \mathcal{T}(M) & \mathcal{T}((M)\ N) := \langle \mathcal{T}(M) \rangle\ \mathcal{T}(N)^{!} \\
\mathcal{T}(0) := \emptyset & \mathcal{T}(a.M) := \mathcal{T}(M) & \mathcal{T}(M + N) := \mathcal{T}(M) \cup \mathcal{T}(N).
\end{array}
$$

$$\frac{s \Rightarrow_\partial \sigma' \quad \bar{t} \Rightarrow_\partial \bar{\tau}'}{\langle \lambda x\, s\rangle\ \bar{t} \Rightarrow_\partial \partial_x \sigma' \cdot \bar{\tau}'} \qquad \frac{}{x \Rightarrow_\partial x} \qquad \frac{s \Rightarrow_\partial \sigma'}{\lambda x\, s \Rightarrow_\partial \lambda x\, \sigma'} \qquad \frac{s \Rightarrow_\partial \sigma' \quad \bar{t} \Rightarrow_\partial \bar{\tau}'}{\langle s\rangle\ \bar{t} \Rightarrow_\partial \langle \sigma'\rangle\ \bar{\tau}'} \qquad \frac{s \Rightarrow_\partial \sigma' \quad \bar{t} \Rightarrow_\partial \bar{\tau}'}{[s] \cdot \bar{t} \Rightarrow_\partial [\sigma'] \cdot \bar{\tau}'}$$

Figure 3 Rules of parallel resource reduction $\Rightarrow_\partial$.

Observe that $\mathcal{T}(M)$ is preserved under $\simeq_+$ so it is well defined on algebraic terms. Also observe that the inclusion $|\Theta(M)| \subseteq \mathcal{T}(M)$ might be strict. Moreover, the identity $\mathcal{T}(M[N/x]) = \mathcal{T}(M)[\mathcal{T}(N)/x]$ is established much more easily than Lemma 17.

We will make crucial use of the Taylor support operator in our study of Taylor expansion: although the latter is our subject of interest, the former is more robust where β-reduction is involved. For instance, an algebraic λ-term M is β-normal iff $\mathcal{T}(M)$ contains only normal resource terms: this fails for $|\Theta(M)|$ (and would also fail had we set $\mathcal{T}(0.N) = \emptyset$).

5 On the reduction of resource vectors

Observe that $\Theta((\lambda x\, M)\ N) = \langle \lambda x\, \Theta(M)\rangle\ \Theta(N)^!$ and $\Theta(M[N/x]) = \partial_x \Theta(M) \cdot \Theta(N)^!$. In order to simulate β-reduction through Taylor expansion we may consider the reduction given by $\epsilon \to \epsilon'$ iff $\epsilon = \sum_{i\in I} a_i.e_i$ and $\epsilon' = \sum_{i\in I} a_i.\epsilon'_i$ with $e_i \to_\partial^? \epsilon'_i$ for all $i \in I$. Observe indeed that, as soon as $(a_i.e_i)_{i\in I}$ is summable, the family $(a_i.\epsilon'_i)_{i\in I}$ is summable too, by Lemma 8. So we do not need any additional condition for this reduction step to be well defined.

This is not enough for simulating β-reduction, though, because we might need to reduce arbitrarily many redexes in parallel:

▶ **Example 20.** Observe that

$$\Theta((x)\ (\lambda y\, y)\ z) = \sum_{n,k_1,\dots,k_n \in \mathbf{N}} \frac{1}{n!k_1!\cdots k_n!} \cdot \langle x\rangle\ [\langle \lambda y\, y\rangle\ z^{k_1}, \dots, \langle \lambda y\, y\rangle\ z^{k_n}]$$

and $\Theta((x)\ z) = \sum_{n\in\mathbf{N}} \frac{1}{n!} \cdot \langle x\rangle\ z^n$. Intuitively, to simulate the β-reduction step $(x)\ (\lambda y\, y)\ z \to_\beta (x)\ z$, we proceed as follows: for each $n, k_1, \dots, k_n$,

- if $k_1 = \cdots = k_n = 1$, we reduce either each $\langle \lambda y\, y\rangle\ z^{k_j} = \langle \lambda y\, y\rangle\ [z]$ to z in parallel;
- otherwise $k_j \neq 1$ for some j, and we $\Rightarrow_\partial$-reduce $\langle \lambda y\, y\rangle\ z^{k_j}$ to 0.

▶ **Definition 21.** We define *parallel resource reduction* $\Rightarrow_\partial \subseteq (!)\Delta \times \mathbf{N}[(!)\Delta]$ by the rules of Fig. 3. We extend it to finite sums: $\sum_{i=1}^n e_i \Rightarrow_\partial \sum_{i=1}^n \epsilon'_i$ whenever $e_i \Rightarrow_\partial \epsilon'_i$ for $1 \leq i \leq n$.

Then $\to_\partial \subseteq \Rightarrow_\partial \subset \to_\partial^*$. Moreover, parallel resource reduction is confluent in a strong sense:

▶ **Definition 22.** We define the *full parallel reduct* $\mathsf{F}(e)$ of $e \in (!)\Delta$ inductively as follows:

$$\mathsf{F}(x) := x \qquad \mathsf{F}(\langle \lambda x\, s\rangle\ \bar{t}) := \partial_x \mathsf{F}(s) \cdot \mathsf{F}(\bar{t}) \qquad \mathsf{F}([s_1, \dots, s_n]) := [\mathsf{F}(s_1), \dots, \mathsf{F}(s_n)]$$
$$\mathsf{F}(\lambda x\, s) := \lambda x\, \mathsf{F}(s) \qquad \mathsf{F}(\langle u\rangle\ \bar{t}) := \langle \mathsf{F}(u)\rangle\ \mathsf{F}(\bar{t}) \quad \text{(if } u \text{ is not an abstraction)}$$

We extend F to sums, setting $\mathsf{F}(\sum_{i=1}^n e_i) = \sum_{i=1}^n \mathsf{F}(e_i)$. We obtain:

▶ **Lemma 23.** *For all $\epsilon, \epsilon' \in \mathbf{N}[(!)\Delta]$, if $\epsilon \Rightarrow_\partial \epsilon'$ then $\epsilon' \Rightarrow_\partial \mathsf{F}(\epsilon)$.*

Parallel reduction $\Rightarrow_\partial$ (like iterated reduction $\to_\partial^*$) lacks the combinatorial regularity properties of $\to_\partial$ given by Lemma 8: $e' \in (!)\Delta$ being fixed, there is no bound on the size of the $\Rightarrow_\partial$-antecedents of e', *i.e.* those $e \in (!)\Delta$ such that $e' \in |\epsilon'|$ with $e \Rightarrow_\partial \epsilon'$.

▶ **Example 24.** Fix $s \in \Delta$. Consider the sequences of resource terms given by $u_0(s) = v_0(s) = s$, $u_{n+1}(s) = \langle \lambda y\, y \rangle\ [u_n(s)]$ and $v_{n+1}(s) = \langle \lambda y\, v_n(s) \rangle\ []$. Then $u_n(s) \Rightarrow_\partial u_{n'}(s)$ and $v_n(s) \Rightarrow_\partial v_{n'}(s)$ for all $n \geq n' \in \mathbf{N}$. In particular $u_n(s) \Rightarrow_\partial s$ and $v_n(s) \Rightarrow_\partial s$.

It is thus no longer automatically possible to reduce all resource expressions in a resource vector simultaneously: consider, *e.g.*, $\sum_{n\in\mathbf{N}} u_n(x)$. Hence, in order to introduce a reduction relation on resource vectors by extending a reduction relation on resource expressions as above, we must in general impose the summability of the family of reducts as a side condition:

▶ **Definition 25.** Fix a relation $\to \subseteq (!)\Delta \times \mathbf{N}[(!)\Delta]$. For all $\epsilon, \epsilon' \in \mathbf{S}^{(!)\Delta}$, we write $\epsilon \mathrel{\widetilde{\to}} \epsilon'$ whenever there exist a family $(a_i)_{i\in I} \in \mathbf{S}^I$, and summable families $(e_i)_{i\in I} \in (!)\Delta^I$ and $(\epsilon'_i)_{i\in I} \in \mathbf{N}[(!)\Delta]^I$ such that: $\epsilon = \sum_{i\in I} a_i.e_i$, $\epsilon' = \sum_{i\in I} a_i.\epsilon'_i$ and for all $i \in I$, $e_i \to^? \epsilon'_i$.

The necessity of such a side condition forbids confluence: *e.g.*, setting $\sigma = \sum_{n\in\mathbf{N}} u_n(v_n(x))$, we have $\sigma \mathrel{\widetilde{\Rightarrow_\partial}} \sum_{n\in\mathbf{N}} u_n(x)$ and $\sigma \mathrel{\widetilde{\Rightarrow_\partial}} \sum_{n\in\mathbf{N}} v_n(x)$, but since the only common reduct of $u_p(x)$ and $v_q(x)$ is x, there is no way to close this pair of reductions in general.

Notice that the families of terms we use in Example 24 involve chains of nested redexes, of unbounded length. The length of such chains in $e \in (!)\Delta$ is moreover bounded by $\mathbf{h}(e)$. Indeed, the size collapse induced by $\Rightarrow_\partial$ is essentially controlled by the height of expressions:

▶ **Lemma 26.** *If $e \Rightarrow_\partial \epsilon'$ and $e' \in |\epsilon'|$ then $\mathbf{s}(e) \leq 4^{\mathbf{h}(e)}\mathbf{s}(e')$.*

Proof. By induction on the reduction $e \Rightarrow_\partial \epsilon'$, using Lemma 6 in the redex case. ◀

Hence we can limit the size collapse if the reduction involves expressions of bounded height. We say $\epsilon \in \mathbf{S}^{(!)\Delta}$ is *bounded* if $\{\mathbf{h}(e)\ ;\ e \in |\epsilon|\}$ is finite. We then write $\mathbf{h}(\epsilon) = \max\{\mathbf{h}(e)\ ;\ e \in |\epsilon|\}$. Observe that $\Theta(M)$ is bounded for all $M \in \Lambda_{\mathbf{S}}$. For all $h \in \mathbf{N}$, we write $(!)\Delta_h := \{e \in (!)\Delta\ ;\ \mathbf{h}(e) \leq h\}$ and then $(!)\mathfrak{B} := \bigcup_{h\in\mathbf{N}} \mathfrak{P}((!)\Delta_h)$ is a finiteness structure: $(!)\mathfrak{B} = \{(!)\Delta_h\ ;\ h \in \mathbf{N}\}^{\perp\perp}$. The semimodule of bounded resource vectors is then $\mathbf{S}\langle(!)\mathfrak{B}\rangle$. We can moreover show that reducing expressions of bounded height generates expressions of bounded height (with a greater bound):

▶ **Lemma 27.** *If $e \Rightarrow_\partial \epsilon'$ and $e' \in |\epsilon'|$ then $\mathbf{h}(e') \leq 2^{\mathbf{h}(e)}\mathbf{h}(e)$.*

▶ **Corollary 28.** *If $(e_i)_{i\in I} \in (!)\Delta_h^I$ is summable and $e_i \Rightarrow_\partial \epsilon'_i$ for all $i \in I$, then $(\epsilon'_i)_{i\in I}$ is summable and $\epsilon'_i \in \mathbf{N}[(!)\Delta_{2^h h}]$ for all $i \in I$.*

In particular, for a fixed $h \in \mathbf{N}$, $(\mathsf{F}(e))_{e\in(!)\Delta_h}$ is summable. Hence F extends to a linear-continuous map $\mathsf{F} : \mathbf{S}^{(!)\Delta_h} \to \mathbf{S}^{(!)\Delta_{2^h h}}$, and more generally to a map $\mathsf{F} : \mathbf{S}\langle(!)\mathfrak{B}\rangle \to \mathbf{S}\langle(!)\mathfrak{B}\rangle$.

Bounded vectors, however, may not be preserved under $\mathrel{\widetilde{\Rightarrow_\partial}}$. Indeed, like in the algebraic λ-calculus, reduction can interact with the semimodule structure of $\mathbf{S}^{(!)\Delta}$: we can reproduce Example 18 in $\mathbf{S}^{(!)\Delta}$ through Taylor expansion (see Corollary 34). More directly, we can use the terms of Example 24: for any $s \in \Delta$, setting $\sigma = \sum_{n\in\mathbf{N}} u_{n+1}(s) \in \mathbf{Z}^\Delta$ we obtain $0 = \sigma - \sigma \mathrel{\widetilde{\Rightarrow_\partial}} \sum_{n\in\mathbf{N}} u_n(s) - \sigma = s$. Of course, this kind of issue does not arise when the semiring of coefficients is *zerosumfree*, *i.e.*, for all $a, b \in \mathbf{S}$, $a + b = 0$ iff $a = b = 0$:

▶ **Lemma 29.** *Assume $\mathbf{S}$ is zerosumfree and fix a relation $\to \subseteq (!)\Delta \times \mathbf{N}[(!)\Delta]$. If $\epsilon \mathrel{\widetilde{\to}} \epsilon'$ then, for all $e' \in |\epsilon'|$ there exists $e \in |\epsilon|$ and $\epsilon_0 \in \mathbf{N}[(!)\Delta]$ such that $e \to^? \epsilon_0$ and $e' \in |\epsilon_0|$.*

Various approaches to get rid of this restriction in the setting of the algebraic λ-calculus [24, 2, 1, 7] may be adapted to the reduction of resource vectors. The linear-continuity of the resource λ-calculus allows us to propose a novel approach: consider possible restrictions on the families of resource expressions simultaneously reduced in a $\mathrel{\widetilde{\to}}$-step.

$$\frac{S \Rightarrow_\beta M' \quad N \Rightarrow_\beta N'}{(\lambda x\, S)\ N \Rightarrow_\beta M'[N'/x]} \qquad \frac{}{x \Rightarrow_\beta x} \qquad \frac{S \Rightarrow_\beta M'}{\lambda x\, S \Rightarrow_\beta \lambda x\, M'} \qquad \frac{S \Rightarrow_\beta M' \quad N \Rightarrow_\beta N'}{(S)\ N \Rightarrow_\beta (M')\ N'}$$

$$\frac{}{0 \Rightarrow_\beta 0} \qquad \frac{M \Rightarrow_\beta M'}{a.M \Rightarrow_\beta a.M'} \qquad \frac{M \Rightarrow_\beta M' \quad N \Rightarrow_\beta N'}{M + N \Rightarrow_\beta M' + N'}$$

Figure 4 Rules of parallel β-reduction $\Rightarrow_\beta$ of algebraic λ-terms.

▶ **Definition 30.** Fix a relation $\to \subseteq (!)\Delta \times \mathbf{N}[(!)\Delta]$. If $\mathcal{E} \subseteq (!)\Delta$, we write $e \to_{|\mathcal{E}} \epsilon'$ iff $e \to \epsilon'$ and $e \in \mathcal{E}$, and then $\widetilde{\to}_{\mathcal{E}} := \widetilde{\to_{|\mathcal{E}}}$. If $\mathfrak{E} \subseteq \mathfrak{P}((!)\Delta)$, we then write $\widetilde{\to}_{\mathfrak{E}}$ for $\bigcup_{\mathcal{E}\in\mathfrak{E}} \widetilde{\to}_{\mathcal{E}}$.

In particular, we have $\widetilde{\to}_{(!)\Delta} = \widetilde{\to}$ and $\widetilde{\to}_{\mathcal{E}} \subseteq \widetilde{\to} \cap (\mathbf{S}^{\mathcal{E}} \times \mathbf{S}^{(!)\Delta})$, but in general the reverse inclusion holds only if $\mathbf{S}$ is zerosumfree: in this latter case $\epsilon \mathrel{\widetilde{\Rightarrow}_\partial} \epsilon'$ iff $\epsilon \mathrel{\widetilde{\Rightarrow}_{\partial|\epsilon|}} \epsilon'$.

We call *reduction structure* any set $\mathfrak{E} \subseteq \mathfrak{P}((!)\Delta)$ such that: (i) $\mathfrak{P}_f((!)\Delta) \subseteq \mathfrak{E}$; (ii) $\mathfrak{E}$ is closed under finite unions; (iii) $\mathfrak{E}$ is downwards closed for inclusion; (iv) for all $\mathcal{E} \in \mathfrak{E}$, $\{e' \in (!)\Delta\ ;\ e' \in |\epsilon'| \text{ with } e \in \mathcal{E} \text{ and } e \to \epsilon'\} \in \mathfrak{E}$. Observe that the first three requirements are automatically satisfied by finiteness structures.

It follows from Lemma 27 that $(!)\mathfrak{B}$ is a $\Rightarrow_\partial$-reduction structure. Let us call *bounded reduction structure* any $\Rightarrow_\partial$-reduction structure $\mathfrak{E}$ such that $\mathfrak{E} \subseteq (!)\mathfrak{B}$. Lemmas 23 and 26 then entail the strong confluence of $\widetilde{\Rightarrow}_{\partial\mathfrak{E}}$:

▶ **Theorem 31.** *For any bounded reduction structure $\mathfrak{E}$, if $\epsilon \mathrel{\widetilde{\Rightarrow}_{\partial\mathfrak{E}}} \epsilon'$ then $\epsilon' \mathrel{\widetilde{\Rightarrow}_{\partial\mathfrak{E}}} \mathsf{F}(\epsilon)$.*

6 Taylor expansion and β-reduction

From now on, for all $M, N \in \Lambda_{\mathbf{S}}$, we write $M \mathrel{\widetilde{\Rightarrow}_\partial} N$ if $\Theta(M) \mathrel{\widetilde{\Rightarrow}_\partial} \Theta(N)$. More generally, for all $M \in \Lambda_{\mathbf{S}}$ and all $\sigma \in \mathbf{S}^{(!)\Delta}$, we write $M \mathrel{\widetilde{\Rightarrow}_\partial} \sigma$ (resp. $\sigma \mathrel{\widetilde{\Rightarrow}_\partial} M$) if $\Theta(M) \mathrel{\widetilde{\Rightarrow}_\partial} \sigma$ (resp. $\sigma \mathrel{\widetilde{\Rightarrow}_\partial} \Theta(M)$). We show that $M \mathrel{\widetilde{\Rightarrow}_\partial} N$ as soon as $M \Rightarrow_\beta N$ where $\Rightarrow_\beta \subseteq \Lambda_{\mathbf{S}} \times \Lambda_{\mathbf{S}}$ is the parallel β-reduction inductively defined on algebraic λ-terms by the rules of Fig. 4.

▶ **Lemma 32.** *If $\sigma \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{S}}} \sigma'$ and $\tau \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{T}}} \tau'$ then we have:*

$$\langle \lambda x\, \sigma \rangle\ \tau \mathrel{\widetilde{\Rightarrow}_{\partial\langle \lambda x\, \mathcal{S}\rangle\mathcal{T}}} \partial_x \sigma' \cdot \tau' \qquad \lambda x\, \sigma \mathrel{\widetilde{\Rightarrow}_{\partial\lambda x\, \mathcal{S}}} \lambda x\, \sigma' \qquad \langle \sigma \rangle\ \tau \mathrel{\widetilde{\Rightarrow}_{\partial\langle \mathcal{S}\rangle\mathcal{T}}} \langle \sigma' \rangle\ \tau' \qquad \sigma^! \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{S}^!}} \sigma'^!.$$

Moreover, if $\epsilon \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{E}}} \epsilon'$ and $\phi \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{F}}} \phi'$ then $a.\epsilon \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{E}}} a.\epsilon'$ and $\epsilon + \phi \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{E}\cup\mathcal{F}}} \epsilon + \phi'$.

Proof. By the multilinear-continuity of syntactic constructs (for $\sigma^! \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{S}^!}} \sigma'^!$ we first prove $\sigma^n \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{S}^!}} \sigma'^n$ for each $n \in \mathbf{N}$) and the fact that summable families form an $\mathbf{S}$-semimodule. ◀

▶ **Theorem 33.** *If $M \Rightarrow_\beta M'$ then $M \mathrel{\widetilde{\Rightarrow}_{\partial\mathcal{T}(M)}} M'$.*

Proof. By induction on the reduction $M \Rightarrow_\beta M'$ using Lemma 32 in each case. ◀

▶ **Corollary 34.** *If $M \Rightarrow_\beta M'$ then $M \mathrel{\widetilde{\Rightarrow}_{\partial(!)\mathfrak{B}}} M'$.*

In particular, if $1 \in \mathbf{S}$ admits an opposite element $-1 \in \mathbf{S}$ then $\widetilde{\Rightarrow}_{\partial(!)\mathfrak{B}}$ is degenerate. Indeed, we can consider $\Rightarrow_\beta$ up to the equality of vector λ-terms by setting $M \mathrel{\widetilde{\Rightarrow}_\beta} N$ if there are $M' \simeq_{\mathrm{v}} M$ and $N' \simeq_{\mathrm{v}} N$ such that $M' \Rightarrow_\beta N'$. Since $\simeq_\Theta$ subsumes $\simeq_{\mathrm{v}}$, we have $M \mathrel{\widetilde{\Rightarrow}_{\partial(!)\mathfrak{B}}} N$ as soon as $M \mathrel{\widetilde{\Rightarrow}_\beta} N$. If $-1 \in \mathbf{S}$, we have $M \mathrel{\widetilde{\Rightarrow}^*_\beta} N$ for all $M, N \in \Lambda_{\mathbf{S}}$ by Example 18, hence $M \mathrel{\widetilde{\Rightarrow}^*_{\partial(!)\mathfrak{B}}} N$.

Even assuming $\mathbf{S}$ is zerosumfree, Taylor expansions are not stable under $\widetilde{\Rightarrow}_\partial$: if $M \mathrel{\widetilde{\Rightarrow}_{\partial\mathfrak{B}}} \sigma'$ there is no reason why σ' would be the Taylor expansion of a term. We do know, however, that $\sigma' \mathrel{\widetilde{\Rightarrow}_{\partial\mathfrak{B}}} \mathsf{F}(\Theta(M))$ and $M \mathrel{\widetilde{\Rightarrow}_{\partial\mathfrak{B}}} \mathsf{F}(\Theta(M))$: we show the latter is the image of a $\Rightarrow_\beta$-step.

▶ **Definition 35.** We define the *full parallel reduct* $\mathsf{F}(S)$ of a simple term S, resp. $\mathsf{F}(M)$ of an algebraic term M, by mutual induction. On algebraic terms, we set $\mathsf{F}(0) := 0$, $\mathsf{F}(a.M) := a.\mathsf{F}(M)$, $\mathsf{F}(M+N) := \mathsf{F}(M) + \mathsf{F}(N)$, On simple terms:

$$\mathsf{F}(x) := x \qquad \mathsf{F}((\lambda x\, S)\ N) := \mathsf{F}(S)[\mathsf{F}(N)/x]$$
$$\mathsf{F}(\lambda x\, S) := \lambda x\, \mathsf{F}(S) \qquad \mathsf{F}((T)\ N) := (\mathsf{F}(T))\ \mathsf{F}(N) \quad \text{(if } T \text{ is not an abstraction).}$$

▶ **Lemma 36.** *If $M \Rightarrow_\beta M'$ then $M' \Rightarrow_\beta \mathsf{F}(M)$. Moreover, for all $M \in \Lambda_{\mathbf{S}}$, $\mathsf{F}(\Theta(M)) = \Theta(\mathsf{F}(M))$.*

Proof. The reduction $M' \Rightarrow_\beta \mathsf{F}(M)$ is constructed by induction on $M \Rightarrow_\beta M'$. The identity $\mathsf{F}(\Theta(M)) = \Theta(\mathsf{F}(M))$ is proved by induction on M: each case is similar to Lemma 32. ◀

▶ **Lemma 37.** *For all bounded reduction structure $\mathfrak{S}$, if $M \mathrel{\widetilde{\Rightarrow}}_{\partial\mathfrak{S}} \sigma'$ then $\sigma' \mathrel{\widetilde{\Rightarrow}}_{\partial\mathfrak{S}} \mathsf{F}(M)$.*

Proof. By Theorem 31, $\sigma' \mathrel{\widetilde{\Rightarrow}}_{\partial\mathfrak{S}} \mathsf{F}(\Theta(M))$ and we conclude by the previous lemma. ◀

▶ **Corollary 38.** *For all bounded reduction structure $\mathfrak{S}$, if $M \mathrel{\widetilde{\Rightarrow}}^n_{\partial\mathfrak{S}} \sigma'$ then $\sigma' \mathrel{\widetilde{\Rightarrow}}^n_{\partial\mathfrak{S}} \mathsf{F}^n(M)$.*

We write $\simeq_\beta$ for the symmetric, reflexive and transitive closure of $\Rightarrow_\beta$. Similarly, if $\mathfrak{E}$ is a bounded reduction structure, we write $\simeq_{\partial\,\mathfrak{E}}$ for the induced equivalence on $\mathbf{S}\langle\mathfrak{E}\rangle$: it follows from Theorem 31 and Corollary 38 that $\epsilon \simeq_{\partial\,\mathfrak{E}} \epsilon'$ iff $\epsilon' \mathrel{\widetilde{\Rightarrow}}^*_{\partial\mathfrak{E}} \mathsf{F}^n(\epsilon)$ for some $n \in \mathbf{N}$. If $M \in \Lambda_{\mathbf{S}}$ is normalizable, we write $\mathsf{NF}(M)$ for its normal form and obtain that, for all N such that $M \simeq_{\partial\,\mathfrak{B}} N$, we have $\mathsf{NF}(M) \mathrel{\widetilde{\Rightarrow}}_{\partial\mathfrak{B}} \mathsf{F}^n(N)$ for some $n \in \mathbf{N}$. In particular, if $\mathbf{S}$ is zerosumfree, $\mathsf{NF}(M) \simeq_\Theta \mathsf{F}^n(N)$. If moreover $M, N \in \Lambda$, we obtain $M \simeq_\beta N$. The next section will allow us to establish a similar conservativity result, without any assumption on $\mathbf{S}$, at the cost of restricting the reduction relation to normalizable resource vectors.

7 Normalization and Taylor expansion commute

Let us first remark that the function $e \in (!)\Delta \mapsto \mathsf{NF}(e) \in \mathbf{N}[(!)\Delta]$ is not summable.

▶ **Example 39.** Recall the family $u_n(s)$ of terms from Example 24: if s is normal, we have $\mathsf{NF}(u_n(x)) = s$ for all $n \in \mathbf{N}$ and the family $(s)_{n\in\mathbf{N}}$ is obviously not summable. With a bit more work, one can show that there are infinitely many $t \in |\Theta(\infty_x)|$ such that $\mathsf{NF}(t) = x$, so NF is not summable even when restricted to the image of Taylor expansion.

We say $\epsilon \in \mathbf{S}^{(!)\Delta}$ is *normalizable* whenever the family $(\mathsf{NF}(e))_{e\in|\epsilon|}$ is summable. In this case, we write $\mathsf{NF}(\epsilon) := \sum_{e\in(!)\Delta} \epsilon_e.\mathsf{NF}(e)$.

Recall from Section 3 that $e \geq_\partial e'$ iff $e \to^*_\partial \epsilon'$ with $e' \in |\epsilon'|$. If $e \in (!)\Delta$, we write $\uparrow e := \{e' \in (!)\Delta\ ;\ e' \geq_\partial e\}$. Then ϵ is normalizable iff for each normal resource expression e, $|\epsilon| \cap \uparrow e$ is finite: writing $(!)\mathcal{N} = \{e \in (!)\Delta\ ;\ e \text{ is normal}\}$ and $(!)\mathfrak{N} = \{\uparrow e\ ;\ e \in (!)\mathcal{N}\}^\perp$, we obtain that $\mathbf{S}\langle(!)\mathfrak{N}\rangle$ is the finiteness space of normalizable resource vectors (in other words, ϵ is normalizable iff $|\epsilon| \in (!)\mathfrak{N}$). For our study of hereditarily determinable terms in Section 8, it will be useful to decompose $(!)\mathfrak{N}$ into a decreasing sequence of finiteness structures.

▶ **Definition 40.** We define the applicative depth $\mathbf{d}(e) \in \mathbf{N}$ of a resource expression $e \in (!)\Delta$ inductively as follows (assuming $\bar{t} = [t_1, \ldots, t_n]$):

$$\mathbf{d}(x) = 1 \quad \mathbf{d}(\lambda x\, s) = \mathbf{d}(s) \quad \mathbf{d}(\langle s\rangle\ \bar{t}) = \max(\mathbf{d}(s), \mathbf{d}(\bar{t}) + 1) \quad \mathbf{d}(\bar{t}) = \max(\mathbf{d}(t_1), \ldots, \mathbf{d}(t_n)).$$

We write $(!)\mathcal{N}_d = \{e \in (!)\mathcal{N} \; ; \; \mathbf{d}(e) \leq d\}$: $(!)\mathcal{N}_d$ is the set of normal resource expressions of applicative depth at most d, so that $(!)\mathcal{N} = \bigcup_{d\in\mathbf{N}}(!)\mathcal{N}_d$. We then write $(!)\mathfrak{N}_d = \{\uparrow e \; ; \; e \in (!)\mathcal{N}_d\}^\perp$ so that $(!)\mathfrak{N} = \bigcap_{d\in\mathbf{N}}(!)\mathfrak{N}_d$. Each finiteness structure $(!)\mathfrak{N}_d$ is moreover a reduction structure for any reduction relation contained in $\to_\partial^*$ (and so is $(!)\mathfrak{N}$):

▶ **Lemma 41.** *If $\mathcal{E} \in (!)\mathfrak{N}_d$ then $\downarrow\mathcal{E} := \{e' \in (!)\Delta \; ; \; e \geq_\partial e'$ with $e \in \mathcal{E}\} \in (!)\mathfrak{N}_d$.*

Proof. Let $e'' \in (!)\mathcal{N}_d$ and $e' \in \downarrow\mathcal{E} \cap \uparrow e''$. Necessarily, there is $e \in \mathcal{E}$ such that $e \geq_\partial e'$. Then $e \in \mathcal{E} \cap \uparrow e''$: since $\mathcal{E} \in (!)\mathfrak{N}_d$, there are finitely many possible values for e hence for e'. ◀

▶ **Lemma 42.** *If $\epsilon \in \mathbf{S}\langle(!)\mathfrak{N}\rangle$ and $\epsilon \mathrel{\widetilde{\Rightarrow_{\partial(!)\mathfrak{N}}}} \epsilon'$ then $\epsilon' \in \mathbf{S}\langle(!)\mathfrak{N}\rangle$ and $\mathsf{NF}(\epsilon) = \mathsf{NF}(\epsilon')$.*

Proof. Assume there exists $\mathcal{E} \in (!)\mathfrak{N}$, a family $\overrightarrow{a} = (a_i)_{i\in I} \in \mathbf{S}^I$, and summable families $(e_i)_{i\in I} \in (!)\Delta^I$ and $(\epsilon'_i)_{i\in I} \in \mathbf{N}[(!)\Delta]^I$ such that: $\epsilon = \sum_{i\in I} a_i.e_i$, $\epsilon' = \sum_{i\in I} a_i.\epsilon'_i$ and for all $i \in I$, $e_i \in \mathcal{E}$ and $e_i \mathrel{\widetilde{\Rightarrow_\partial}} \epsilon'_i$. We obtain that $\mathcal{E}' := \bigcup_{i\in I} |\epsilon'_i| \in (!)\mathfrak{N}$ by Lemma 41, hence $\epsilon' \in \mathbf{S}\langle(!)\mathfrak{N}\rangle$ since $|\epsilon'| \subseteq \mathcal{E}'$. Then, by the linear-continuity of NF on $\mathbf{S}^{\mathcal{E}'}$, $\mathsf{NF}(\epsilon) = \sum_{i\in I} a_i.\mathsf{NF}(e_i) = \sum_{i\in I} a_i.\mathsf{NF}(\epsilon'_i) = \mathsf{NF}(\sum_{i\in I} a_i.\epsilon'_i) = \mathsf{NF}(\epsilon')$. ◀

The following three lemmas are the key steps of a reducibility argument: like in Ehrhard's work for the typed case [10], or our previous work for the strongly normalizable case [22], each $(!)\mathfrak{N}_d$ is the analogue of a reducibility candidate. Similar results for $(!)\mathfrak{N}$ are immediately derived from those. In our setting, the proof of Lemma 45 is made easier by the fact that, for $\mathcal{E} \in (!)\mathfrak{N}_d$, we require $\mathcal{E} \cap \uparrow e$ to be finite only when e is normal.

▶ **Lemma 43.** *If $\mathcal{S} \in \mathfrak{N}_d$ then $\lambda x\, \mathcal{S} \in \mathfrak{N}_d$.*

▶ **Lemma 44.** *If $\mathcal{T}_1, \dots, \mathcal{T}_n \in \mathfrak{N}_d$ then $\langle x\rangle \; \mathcal{T}_1^! \cdots \mathcal{T}_n^! \in \mathfrak{N}_{d+1}$.*

▶ **Lemma 45.** *If $\langle \partial_x \mathcal{S} \cdot \overline{\mathcal{T}}_0\rangle \; \overline{\mathcal{T}}_1 \cdots \overline{\mathcal{T}}_n \in \mathfrak{N}_d$ then $\langle \lambda x\, \mathcal{S}\rangle \; \overline{\mathcal{T}}_0 \, \overline{\mathcal{T}}_1 \cdots \overline{\mathcal{T}}_n \in \mathfrak{N}_d$*

Observe that the weak normalizability of algebraic λ-terms is not stable under $\simeq_{\mathrm{v}}$ because of the identity $0 \simeq_{\mathrm{v}} 0.M$. Since we work up to $\simeq_+$ only, a general standardization argument [21] applies and we have that $M \in \Lambda_{\mathbf{S}}$ is normalizable iff the left reduction strategy terminates, *i.e.* there exists $k \in \mathbf{N}$ such that $\mathsf{L}^k(M)$ is normal, where L is defined as follows:

▶ **Definition 46.** We define the *left reduct* $\mathsf{L}(S)$ of a simple term S, resp. $\mathsf{L}(M)$ of an algebraic term M, by mutual induction. On algebraic terms, we set $\mathsf{L}(0) := 0$, $\mathsf{L}(a.M) := a.\mathsf{L}(M)$, $\mathsf{L}(M+N) := \mathsf{L}(M) + \mathsf{L}(N)$, On simple terms:

$$\mathsf{L}(\lambda x\, S) := \lambda x\, \mathsf{L}(S) \qquad \mathsf{L}((x)\; M_1 \cdots M_n) := (x)\; \mathsf{L}(M_1) \cdots \mathsf{L}(M_n)$$
$$\mathsf{L}((\lambda x\, S)\; M_0\, M_1 \cdots M_n) := (S[M_0/x])\; M_1 \cdots M_n$$

▶ **Theorem 47.** *If M is normalizable, then $\mathcal{T}(M) \in \mathfrak{N}$, and $\Theta(M) \in \mathbf{S}\langle\mathfrak{N}\rangle$.*

Proof. By induction on k such that $\mathsf{L}^k(M)$ is normal, then on M. We use Lemmas 43 to 45 for $\mathfrak{N}$ in the case of simple terms, and the fact that $\mathfrak{N}$ is a finiteness structure otherwise. ◀

Writing $\mathsf{NF}(M)$ for the normal form of M, it remains to prove that $\Theta(\mathsf{NF}(M)) = \mathsf{NF}(\Theta(M))$.

▶ **Lemma 48.** *We define the left reduct of a resource expression inductively as follows:*

$$\mathsf{L}(\lambda x\, s) := \lambda x\, \mathsf{L}(s) \qquad \mathsf{L}(\langle x\rangle\; \bar{t}_1 \cdots \bar{t}_n) := \langle x\rangle\; \mathsf{L}(\bar{t}_1) \cdots \mathsf{L}(\bar{t}_n)$$
$$\mathsf{L}([t_1, \dots, t_n]) := [\mathsf{L}(t_1), \dots, \mathsf{L}(t_n)] \qquad \mathsf{L}(\langle \lambda x\, s\rangle\; \bar{t}_0\, \bar{t}_1 \cdots \bar{t}_n) := \langle \partial_x s \cdot \bar{t}_0\rangle\; \bar{t}_1 \cdots \bar{t}_n$$

Then for all resource expression $e \in (!)\Delta$, $e \Rightarrow_\partial \mathsf{L}(e)$. In particular $\mathsf{NF}(e) = \mathsf{NF}(\mathsf{L}(e))$. If moreover $e' \in |\mathsf{L}(e)|$ then $\mathbf{s}(e) \leq 4\mathbf{s}(e')$.

$$\frac{}{M \Downarrow_0} \quad \frac{M \Uparrow}{M \Downarrow_d} \quad \frac{S \Downarrow_d}{\lambda x\, S \Downarrow_d} \quad \frac{M_1 \Downarrow_d \quad \cdots \quad M_n \Downarrow_d}{(x)\ M_1 \cdots M_n \Downarrow_{d+1}} \quad \frac{M \Downarrow_d}{a.M \Downarrow_d} \quad \frac{M \Downarrow_d \quad N \Downarrow_d}{M + N \Downarrow_d} \quad \frac{(S[M_0/x])\ M_1 \cdots M_n \Downarrow_d}{(\lambda x\, S)\ M_0 \cdots M_n \Downarrow_d}$$

Figure 5 Rules for determinable terms.

Proof. Similar to Lemmas 23 and 26. ◀

As a consequence $(\mathsf{L}(e))_{e \in (!)\Delta}$ is summable, so that $\mathsf{L}(\Theta(M)) = \Theta(\mathsf{L}(M))$, for any $M \in \Lambda_{\mathbf{S}}$.

▶ **Theorem 49.** *If M is normalizable, then $\mathsf{NF}(\Theta(M)) = \Theta(\mathsf{NF}(M))$.*

Proof. Assume $\mathsf{NF}(M) = \mathsf{L}^k(M)$. By Theorem 47 and Lemma 42, $\mathsf{L}^k(\Theta(M)) \in \mathbf{S}\langle\mathfrak{N}\rangle$ and $\mathsf{NF}(\Theta(M)) = \mathsf{NF}(\mathsf{L}^k(\Theta(M)))$. Then $\Theta(\mathsf{NF}(M)) = \Theta(\mathsf{L}^k(M)) = \mathsf{L}^k(\Theta(M)) = \mathsf{NF}(\Theta(M))$. ◀

If M and N are normalizable and $M \simeq_{\partial\, \mathfrak{N}\cap\mathfrak{B}} N$, Theorems 31 and 33, and Corollary 38, entail that $\mathsf{L}^k(\mathsf{NF}(M)) \simeq_{\partial\, \mathfrak{N}\cap\mathfrak{B}} \mathsf{NF}(N)$ for some $k \in \mathbf{N}$. Since $\mathsf{L}^k(\mathsf{NF}(M)) = \mathsf{NF}(M)$, the previous Theorem entails $\mathsf{NF}(M) \simeq_\Theta \mathsf{NF}(N)$. In particular if M and N are pure λ-terms, we have $M \simeq_\beta N$. It follows that $\simeq_{\partial\, \mathfrak{N}\cap\mathfrak{B}}$ is a non trivial extension of $\simeq_\beta$ on normalizable algebraic λ-terms. We can extend this conservativity result to non-normalizing pure λ-terms thanks to previous work by Ehrhard and Regnier:

▶ **Theorem 50** ([13, 12])**.** *For all pure λ-term $M \in \Lambda$, $\mathcal{T}(M) \in \mathfrak{N}$ and $\mathsf{NF}(\Theta(M)) = \Theta(\mathsf{BT}(M))$ where $\mathsf{BT}(M)$ denotes the Böhm tree of M.*

▶ **Lemma 51.** *If $M, N \in \Lambda$ and $M \simeq_{\partial\, \mathfrak{N}\cap\mathfrak{B}} N$ then $\mathsf{BT}(M) = \mathsf{BT}(N)$.*

Proof. By confluence, there is σ such that $M \widetilde{\Rightarrow}^*_{\partial\, \mathfrak{N}\cap\mathfrak{B}} \sigma$ and $N \widetilde{\Rightarrow}^*_{\partial\, \mathfrak{N}\cap\mathfrak{B}} \sigma$, and by Theorems 49 and 50 we obtain: $\Theta(\mathsf{BT}(M)) = \mathsf{NF}(\Theta(M)) = \mathsf{NF}(\sigma) = \mathsf{NF}(\Theta(N)) = \Theta(\mathsf{BT}(N))$. We conclude since Θ is injective on Böhm trees. ◀

8 Normal form of Taylor expansion, *façon* Böhm trees

In the ordinary λ-calculus, head normalizable terms are exactly those with a non trivial Böhm tree. This is reflected via Taylor expansion: $\mathsf{NF}(\Theta(M)) = 0$ iff M has no head normal form. In the non-uniform case, we take this characterization as a definition: we say an algebraic λ-term $M \in \Lambda_{\mathbf{S}}$ is *Taylor-unsolvable* and write $M \Uparrow$ if $\mathsf{NF}(s) = 0$ for all $s \in \mathcal{T}(M)$.

▶ **Definition 52.** Let $M \in \Lambda_{\mathbf{S}}$ be an algebraic λ-term. We say M is *d-determinable* if the judgement $M \Downarrow_d$ can be derived inductively from the rules of Fig. 5. We say M is *hereditarily determinable* and write $M \Downarrow_\omega$ if $M \Downarrow_d$ for all $d \in \mathbf{N}$. We say M is in *d-determinate form* and write $M\, \mathsf{df}_d$ if $M \Downarrow_d$ is derivable from the above rules excluding the last one (head redex).

We have $M \Downarrow_\omega$ whenever M is normalizable. Moreover, $M \Downarrow_\omega$ for all $M \in \Lambda$.

▶ **Lemma 53.** *If $M \Downarrow_d$ then $\mathcal{T}(M) \in \mathfrak{N}_d$.*

Proof. By induction on the derivation of $M \Downarrow_d$, using in particular Lemmas 43 to 45. ◀

▶ **Lemma 54.** *If $M\, \mathsf{df}_d$, $s \in \mathcal{T}(M)$ and $s \geq_\partial s'$ with $s' \in \mathcal{N}_d$, then $s = s'$. If moreover $M \Rightarrow_\beta M'$ then $\Theta(M)_{s'} = \Theta(M')_{s'}$.*

Proof. By induction on the reduction $s \to^*_\partial \sigma'$ with $s' \in |\sigma'|$. ◀

▶ **Theorem 55.** *If $M \Downarrow_\omega$ then $\mathcal{T}(M) \in \mathfrak{N}$. Moreover, if $s \in \mathcal{N}_d$ then for all M' such that $M \simeq_\beta M'$ and $M'\, \mathsf{df}_d$ we have $\mathsf{NF}(\Theta(M))_s = \Theta(M')_s$.*

Proof. The fact that $\mathcal{T}(M) \in \mathfrak{N}$ follows from Lemma 53. It is easy to prove by induction on d that there exists $k \in \mathbf{N}$ such that $\mathsf{L}^k(M)\,\mathsf{df}_d$. Let $s \in \mathcal{N}_d$: we first prove $\mathsf{NF}(\Theta(M))_s = \Theta(\mathsf{L}^k(M))_s$. Since $M \Rightarrow_\beta^* \mathsf{L}^k(M)$, Theorem 33 and Lemma 42 entail $\mathsf{NF}(\Theta(M)) = \mathsf{NF}(\Theta(\mathsf{L}^k(M)))$. We then obtain $\mathsf{NF}(\Theta(\mathsf{L}^k(M)))_s = \Theta(\mathsf{L}^k(M))_s$, using Lemma 54. Now assume $M \simeq_\beta M'$ and $M'\,\mathsf{df}_d$. By the confluence of $\Rightarrow_\beta$, there exists $M'' \in \Lambda_{\mathbf{S}}$ such that $\mathsf{L}^k(M) \Rightarrow_\beta^* M''$ and $M' \Rightarrow_\beta^* M''$: we conclude by Lemma 54 again. ◀

The d-determinate forms of an hereditarily determinable term M differ only by subterms that are Taylor-unsolvable or at applicative depth greater than d. If we cut out unsolvable subterms, and replace the arguments of applications at depth d by 0, the common value is a normal term M_d with $\mathbf{d}(M_d) \leq d$: this is exactly the analogue of a Böhm tree approximant. In particular, Theorem 55 is a proper generalization of Theorem 50, and hereditarily determinable terms form the maximal class for which such an extension may hold: without any topological information on scalars, nothing can be said about infinite sums of head normal forms.

9 Conclusion

We have presented a notion of reduction on resource vectors, simulating β-reduction, and proved that Taylor expansion and normalization commute in a generic non-uniform setting. In the final section we have outlined the basic blocks of a generalization of Böhm trees. This construction is essentially *ad-hoc*: its only purpose is to generalize Theorem 50. Our main achievement here is conceptual: we have formalized the idea that *the normal form of Taylor expansion is an appropriate notion of denotation in a non-uniform setting.*

A natural question to ask is how normalizing the Taylor expansion compares with pre-existing notions of denotation in various non-deterministic settings: must-non-deterministic Böhm trees [6], probabilistic Böhm trees [21], weighted relational models [4, 20, 19], *etc.* Even closer to our work, Tsukada, Asada and Ong have recently established [23] the commutation between computing Böhm trees of non-deterministic λ-terms and Taylor expansion with coefficients taken in the complete semiring of positive reals, where all sums converge.

Their approach is based on an precise description of the relationship between the coefficients of resource terms in the expansion of a term and those in the expansion of its Böhm-tree, using a *rigid* Taylor expansion as an intermediate step: this avoids the ambiguity between the sums of coefficients generated by redundancies in the expansion and those representing non-deterministic superpositions. Their work can thus be considered as a refinement of Ehrhard and Regnier's method, that they are moreover able to generalize to the non-deterministic case provided the semiring of scalars is complete. By contrast, our approach is focused on β-reduction and identifies a class of algebraic λ-terms for which the normalization of Taylor expansion converges independently from the topology on scalars. It seems only natural to study the connections between both approaches, in particular to tackle the case of weighted non-determinism in a complete semiring, as a first step towards the treatment of probabilistic or quantum superposition.

We are moreover investigating the extension of our approach to linear logic proof nets, as well as to infinitary λ-calculus [18]: the Taylor expansion operator extends naturally to both proof nets and infinitary λ-terms, and preliminary work suggests that our study of reduction under Taylor expansion is robust enough that it could be adapted to both settings.

Acknowledgments. The author wishes to thank all three referees for providing relevant remarks and valuable advice for preparing the final version of this article.

References

1 M. Alberti. *On operational properties of quantitative extensions of λ-calculus*. PhD thesis, Aix Marseille Université and Università di Bologna, 2014. URL: https://hal.inria.fr/tel-01096067.

2 P. Arrighi and G. Dowek. Linear-algebraic lambda-calculus: higher-order, encodings, and confluence. In *RTA 2008*, volume 5117 of *LNCS*. Springer, 2008. URL: http://dx.doi.org/10.1007/978-3-540-70590-1_2, doi:10.1007/978-3-540-70590-1_2.

3 A. Bucciarelli, T. Ehrhard, and G. Manzonetto. Not enough points is enough. In *CSL 2007*, volume 4646 of *LNCS*. Springer Berlin, 2007.

4 V. Danos and T. Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation*, 2011. doi:10.1016/j.ic.2011.02.001.

5 D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *Math. Struct. in Comp. Sci.*, 2017.

6 U. de' Liguoro and A. Piperno. Nondeterministic extensions of untyped λ-calculus. *Information and Computation*, 1995.

7 A. Díaz-Caro. *Du typage vectoriel*. PhD thesis, Université de Grenoble, France, 2011.

8 T. Ehrhard. On Köthe sequence spaces and linear logic. *Math. Struct. in Comp. Sci.*, 2001.

9 T. Ehrhard. Finiteness spaces. *Math. Struct. in Comp. Sci.*, 15(4):615–646, 2005.

10 T. Ehrhard. A finiteness structure on resource terms. In *LICS 2010*. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.38.

11 T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theor. Comp. Sci.*, 2003.

12 T. Ehrhard and L. Regnier. Böhm trees, Krivine's machine and the Taylor expansion of λ-terms. In *CiE 2006*, volume 3988 of *LNCS*. Springer, 2006. doi:10.1007/11780342_20.

13 T. Ehrhard and L. Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theor. Comp. Sci.*, 2008. doi:10.1016/j.tcs.2008.06.001.

14 J.-Y. Girard. The system F of variable types, fifteen years later. *Theor. Comp. Sci.*, 1986. doi:10.1016/0304-3975(86)90044-7.

15 J.-Y. Girard. Normal functors, power series and lambda-calculus. *Annals of Pure and Applied Logic*, 1988.

16 R. Hasegawa. The generating functions of lambda terms. In *First Conference of the Centre for Discrete Mathematics and Theoretical Computer Science, Auckland, New Zealand, December, 9-13, 1996*. Springer-Verlag, Singapore, 1996.

17 A. Joyal. Foncteurs analytiques et espèces de structures. In *Proceedings of the "Colloque de combinatoire énumérative", Montréal, May 28 – June 1, 1985*. Springer, 1986. doi:10.1007/BFb0072514.

18 J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. Infinitary lambda calculus. *Theor. Comp. Sci.*, 1997.

19 J. Laird. Fixed points in quantitative semantics. In *LICS 2016*. ACM, 2016. doi:10.1145/2933575.2934569.

20 J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. Weighted relational models of typed lambda-calculi. In *LICS 2013*. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.36.

21 T. Leventis. *Probabilistic λ-theories*. PhD thesis, Aix-Marseille Université, 2016.

22 M. Pagani, C. Tasson, and L. Vaux. Strong normalizability as a finiteness structure via the taylor expansion of λ-terms. In *FOSSACS 2016*, volume 9634 of *LNCS*. Springer, 2016. doi:10.1007/978-3-662-49630-5_24.

23 Takeshi Tsukada, Kazuyuki Asada, and Luke Ong. Generalised species of rigid resource terms. In Joel Ouaknine, editor, *Proceedings of the 32nd Annual IEEE Symposium on Logic in Computer Science, LICS 2017*, June 2017. To appear.

24 L. Vaux. The algebraic lambda calculus. *Math. Struct. in Comp. Sci.*, 2009. doi:10.1017/S0960129509990089.

On the First-Order Complexity of Induced Subgraph Isomorphism*

Oleg Verbitsky[1] and Maksim Zhukovskii[2]

1 **Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany**
2 **Moscow Institute of Physics and Technology, Department of Discrete Mathematics, Moscow, Russia**

Abstract

Given a graph F, let $\mathcal{I}(F)$ be the class of graphs containing F as an induced subgraph. Let $W[F]$ denote the minimum k such that $\mathcal{I}(F)$ is definable in k-variable first-order logic. The recognition problem of $\mathcal{I}(F)$, known as Induced Subgraph Isomorphism (for the pattern graph F), is solvable in time $O(n^{W[F]})$. Motivated by this fact, we are interested in determining or estimating the value of $W[F]$. Using Olariu's characterization of paw-free graphs, we show that $\mathcal{I}(K_3 + e)$ is definable by a first-order sentence of quantifier depth 3, where $K_3 + e$ denotes the paw graph. This provides an example of a graph F with $W[F]$ strictly less than the number of vertices in F. On the other hand, we prove that $W[F] = 4$ for all F on 4 vertices except the paw graph and its complement. If F is a graph on ℓ vertices, we prove a general lower bound $W[F] > (1/2 - o(1))\ell$, where the function in the little-o notation approaches 0 as ℓ increases. This bound holds true even for a related parameter $W^*[F] \leq W[F]$, which is defined as the minimum k such that $\mathcal{I}(F)$ is definable in the infinitary logic $L^k_{\infty\omega}$. We show that $W^*[F]$ can be strictly less than $W[F]$. Specifically, $W^*[P_4] = 3$ for P_4 being the path graph on 4 vertices.

1998 ACM Subject Classification F.4.1 Finite Model Theory

Keywords and phrases The induced subgraph isomorphism problem, descriptive and computational complexity, finite-variable first-order logic, quantifier depth and variable width

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.40

1 Introduction

For a given graph F, let $\mathcal{I}(F)$ denote the class of all graphs containing a copy of F as an induced subgraph. We are interested in the descriptive complexity of $\mathcal{I}(F)$, for a fixed pattern graph F, in first-order logic whose vocabulary consists of the adjacency and the equality relations ($\sim$ and $=$ respectively). Let $D[F]$ denote the minimum quantifier depth of a sentence in this logic that defines $\mathcal{I}(F)$. Furthermore, let $W[F]$ denote the minimum variable width of a sentence defining $\mathcal{I}(F)$. Note that

$$W[F] \leq D[F] \leq \ell,$$

where here and throughout the paper ℓ denotes the number of vertices in F. It may come as some surprise that the parameter $D[F]$ can be strictly less than ℓ. To see an example, let

* O. Verbitsky was supported by DFG grant VE 652/1–2. He is on leave from the Institute for Applied Problems of Mechanics and Mathematics, Lviv, Ukraine. M. Zhukovskii was supported by grants No. 15-01-03530 and 16-31-60052 of Russian Foundation for Basic Research.

$F = K_3 + e$ be the paw graph  . The following sentence uses three variables x_1, x_2, x_3 and has quantifier depth 3:

$$\exists x_1 \, (\exists x_2 \exists x_3 \, (x_1 \sim x_2 \wedge x_1 \sim x_3 \wedge x_2 \sim x_3) \wedge \\ \exists x_2 \, (x_1 \not\sim x_2 \wedge \exists x_3 \, (x_1 \sim x_3 \wedge x_3 \sim x_2) \wedge \exists x_3 \, (x_3 \sim x_1 \wedge x_3 \not\sim x_2))).$$

It says that a graph contains a vertex v that belongs to a triangle and can be accompanied with a vertex u at distance 2 from v such that there is a vertex w adjacent to v but non-adjacent to u. Obviously, this sentence is true on the paw graph and on every graph containing an induced paw subgraph. Olariu's characterization of paw-free graphs [21] implies that the sentence is false on every graph without an induced paw subgraph; see Section 4.1 for details.

The decision problem for $\mathcal{I}(F)$ is known as Induced Subgraph Isomorphism (for the pattern graph F). We denote it by ISI(F). Our interest in the parameters $D[F]$ and $W[F]$ is motivated by the fact that ISI(F) is solvable in time $O(n^{W[F]})$; see [19, Prop. 6.6]. Before stating our results on $D[F]$ and $W[F]$ we very briefly overview the known algorithmic results in this area.

1.1 Computational complexity of Induced Subgraph Isomorphism

Obviously, ISI(F) is solvable in time $O(n^\ell)$ on n-vertex input graphs by exhaustive search. We use the standard notation K_ℓ for complete graphs, P_ℓ for paths, and C_ℓ for cycles on ℓ vertices. Itai and Rodeh [14] observed that ISI(K_3) is solvable in time $O(n^\omega)$, where $\omega < 2.373$ is the exponent of fast square matrix multiplication [11]. Nešetřil and Poljak [20] showed, by a reduction of ISI(F) to ISI(K_3), that ISI(F) is solvable in time $O(n^{(\omega/3)\ell+c})$, where $c = 0$ if ℓ is divisible by 3 and $c \leq \frac{2}{3}$ otherwise. For ℓ not divisible by 3, this time bound was improved by Eisenbrand and Grandoni [8] using fast rectangular matrix multiplication. On the other hand, ISI(K_ℓ) is unsolvable in time $n^{o(\ell)}$ unless the Exponential Time Hypothesis fails [3]. Floderus et al. [10] collected evidence in favour of the conjecture that ISI(F) for F with ℓ vertices cannot be solved faster than ISI($K_{c\ell}$), where $c < 1$ is a constant independent on F. Along with the Exponential Time Hypothesis, this would imply that the time complexity of ISI(F) is $n^{\Theta(\ell)}$. As an example of a particular result of [10] in this direction, ISI(P_{2a-1}) is not easier than ISI(K_a), and the same holds true for ISI(C_{2a}); see also the earlier work [4].

The induced subgraph isomorphism problem has been intensively studied for particular pattern graphs F with small number of vertices. Let $\overline{F}$ denote the complement graph of F and note that at least one of the graphs F and $\overline{F}$ is connected. Since ISI(F) and ISI($\overline{F}$) have the same time complexity, we can restrict our attention to connected pattern graphs. There are six such graphs on four vertices, namely K_4, P_4, C_4, $K_3 + e$, the claw graph $K_{1,3}$ (), and the diamond graph $K_4 \setminus e$ (). Corneil et al. [5] designed an $O(n + m)$ time algorithm for ISI(P_4), where m denotes the number of edges in an input graph. As noted in [17], the Olariu characterization of paw-free graphs reduces ISI($K_3 + e$) to ISI(K_3), showing that the former problem is also solvable in time $O(n^\omega)$. The same time bound is obtained by Vassilevska Williams et al. [28] for the diamond graph $K_4 \setminus e$ and, using randomization, for the other pattern graphs on 4 vertices except K_4. The best known time bound for ISI(K_4) is given by the methods of [8] and is currently $O(n^{3.257})$ [28].

1.2 Our results on the descriptive complexity of Induced Subgraph Isomorphism

In Section 3 we prove a general lower bound $W[F] > (1/2 - o(1))\ell$, where the function in the little-o notation approaches 0 as ℓ, the number of vertices in the pattern graph F, increases.

Whether or not it can be improved remains an intriguing open question. Note that this bound leaves a hypothetical possibility that the time bound $O(n^{W[F]})$ for Induced Graph Isomorphism can be better than the Nešetřil-Poljak bound $O(n^{(\omega/3)\ell+c})$ for infinitely many pattern graphs F.

Our approach uses a connection to the k-extension property of graphs, that is well known in finite model theory; see, e.g, [25]. We define the *extension index* of F, denoted by $E[F]$, as the minimum k such that the k-extension property forces the existence of an induced copy of F. It is easy to show that $W[F] \geq E[F]$. Our results about the parameter $E[F]$ may be interesting on its own. In particular, we show that $E[F] \geq \chi(F)$, where $\chi(F)$ denotes the chromatic number of F.

In Section 4, we determine the values of $D[F]$ and $W[F]$ for all pattern graphs with at most 4 vertices. We prove that $W[F] = 4$ for all F on 4 vertices except the paw graph and its complement.

Our lower bounds for $W[F]$ hold true also for a related parameter $W^*[F] \leq W[F]$, which is defined as the minimum k such that $\mathcal{I}(F)$ is definable in the infinitary logic $L^k_{\infty\omega}$. The only exception is $F = P_4$, the path on 4 vertices. In this case, we prove that $W^*[P_4] = 3$ while $W[P_4] = 4$. This shows that $W^*[F]$ can be strictly less than $W[F]$, that is, the infinitary logic can be more succinct when defining $\mathcal{I}(F)$.

In Section 5, we address a relaxation version of the parameter $W[F]$. Consider a simple example. Let $D_v[F]$ be the minimum quantifier depth of a sentence Φ defining $\mathcal{I}(F)$ over *sufficiently large connected* graphs. That is, it is required that there is a number s such that Φ correctly detects whether or not a graph G belongs to $\mathcal{I}(F)$ only if G is connected and has at least s vertices. Whereas $D_v[F] \leq D[F]$, it is clear that $\mathcal{I}(F)$ for a connected pattern graph F is still recognizable in time $O(n^{D_v[F]})$. Let $F = P_3$. As easily seen, P_3-free graphs are exactly disjoint unions of cliques. Therefore, connected P_3-free graphs are exactly the complete graphs, which readily implies that $D_v[P_3] \leq 2$, whereas $D[P_3] = 3$. As a further example, we remark that the existence of a *not necessarily induced* subgraph P_4 can be defined over sufficiently large connected graphs with just 2 variables; see [26] for this and further examples.

We can go further and define $W_{tw}[F]$ to be the minimum variable width of a sentence defining $\mathcal{I}(F)$ over connected graphs G of *sufficiently large treewidth* $tw(G)$. As a consequence of Courcelle's theorem [7], $\mathcal{I}(F)$ for a connected pattern graph F is recognizable in time $O(n^{W_{tw}[F]})$; cf. the discussion in [26].

The above discussion motivates the problem of proving lower bounds for the parameter $W_\kappa[F]$ which we define as the minimum variable width of a sentence defining $\mathcal{I}(F)$ over graphs G of *sufficiently large connectedness* $\kappa(G)$. Note that $W_\kappa[F] \leq W_{tw}[F] \leq W_v[F] \leq W[F]$. We prove that $W_\kappa[F] = W[F]$ for a large class of pattern graphs F. We also prove a general lower bound $W_\kappa[F] > (1/3 - o(1))\,\ell$ for all F on ℓ vertices.

1.3 Comparison to (not necessarily induced) Subgraph Isomorphism

The Subgraph Isomorphism problem is very different from its induced version. For infinitely many pattern graphs F, Subgraph Isomorphism can be solved in time $O(n^c)$ for a constant c. This follows from a result by Alon, Yuster and Zwick [1] who showed that the problem is solvable in time $2^{O(\ell)} \cdot n^{tw(F)+1} \log n$.

Let $W(F)$ and $D(F)$ be the analogs of $W[F]$ and $D[F]$ for the not-necessarily-induced case. Note that $W(F) = D(F) = \ell$ as a consequence of the trivial observation that K_ℓ contains F as a subgraph while $K_{\ell-1}$ does not. Nevertheless, $D_v[F]$ can be strictly smaller

than ℓ for some connected pattern graphs. Moreover, $D_{tw}[F]$ can sometimes be arbitrarily small comparing to ℓ. This is the subject of our preceding paper [26].

1.4 Logic with numeric predicates

In the present paper, we consider the most laconic first-order language of graphs whose vocabulary has only the adjacency and the equality relations. If we assume that the vertex set of a graph is $\{1, 2, \ldots, n\}$ and additionally allow arbitrary numerical relations, this richer logic captures the non-uniform AC^0; see [13, 19]. Let $W_{\mathrm{Arb}}(F)$ denote the analog of the parameter $W(F)$ (the not-necessarily-induced case) for this logic, and $W_{\mathrm{Arb}}[F]$ denote the analog of the parameter $W[F]$ (the induced case). The known relations to circuit complexity [13, 24] imply that the (not necessarily induced) Subgraph Isomorphism is solvable by bounded-depth unbounded-fan-in circuits of size $n^{W_{\mathrm{Arb}}(F)+o(1)}$, and the similar is true also for Induced Subgraph Isomorphism. Whereas the parameter $W_{\mathrm{Arb}}(F)$ is studied in this context by Li, Razborov, and Rossman [18], the parameter $W_{\mathrm{Arb}}[F]$ remains unexplored.

2 Preliminaries

A *graph property* is a class of graphs $\mathcal{C}$ closed under isomorphism, that is, for isomorphic graphs G and H, $G \in \mathcal{C}$ iff $H \in \mathcal{C}$. We consider first-order sentences about graphs in the language containing the adjacency and the equality relations. A sentence Φ *defines* a graph property $\mathcal{C}$ if $G \in \mathcal{C}$ exactly when $G \models \Phi$, i.e., Φ is true on G. A graph property $\mathcal{C}$ is *first-order definable* if there is a first-order sentence defining $\mathcal{C}$.

Let $\mathcal{C}$ be a first-order graph property. The *logical depth* of $\mathcal{C}$, denoted by $D(\mathcal{C})$, is the minimum quantifier depth (rank) of a sentence defining $\mathcal{C}$. The *logical width* of $\mathcal{C}$, denoted by $W(\mathcal{C})$, is the minimum *variable width* of a sentence defining $\mathcal{C}$, i.e., the number of first-order variables occurring in the sentence where different occurrences of the same variable do not count.

Given two non-isomorphic graphs G and H, let $D(G, H)$ (resp. $W(G, H)$) denote the minimum quantifier depth (resp. variable width) of a sentence that *distinguishes* G and H, i.e., is true on one of the graphs and false on the other.

▶ **Lemma 1.** $D(\mathcal{C}) = \max\{\, D(G, H) : G \in \mathcal{C},\ H \notin \mathcal{C}\}$.

Proof. In one direction, note that whenever $G \in \mathcal{C}$ and $H \notin \mathcal{C}$, we have $D(G, H) \leq D(\mathcal{C})$ because any sentence defining $\mathcal{C}$ distinguishes G and H. For the other direction, suppose that every such G and H are distinguished by a sentence $\Phi_{G,H}$ of quantifier depth at most d. Specifically, suppose that $\Phi_{G,H}$ is true on G and false on H. We have to show that $\mathcal{C}$ can be defined by a sentence of quantifier depth at most d. For a graph $G \in \mathcal{C}$, consider the sentence $\Phi_G \stackrel{\text{def}}{=} \bigwedge_H \Phi_{G,H}$, where the conjunction is over all $H \notin \mathcal{C}$. This sentence distinguishes G from all $H \notin \mathcal{C}$ and has quantifier depth at most d. The only problem with it is that the conjunction over H is actually infinite. Luckily, there are only finitely many pairwise inequivalent first-order sentences about graphs of quantifier depth d; see, e.g., [23, Theorem 2.4]. Removing all but one formula $\Phi_{G,H}$ from each equivalence class, we make Φ_G a legitimate finite sentence. Now, consider $\Phi \stackrel{\text{def}}{=} \bigvee_G \Phi_G$, where the disjunction is over all $G \in \mathcal{C}$. It can be made finite in the same way. The sentence Φ defines $\mathcal{C}$ and has quantifier depth at most d. ◀

Thus, Lemma 1 is a simple consequence of the fact that there are only finitely many pairwise inequivalent first-order statements of bounded quantifier depth. Note that the last

fact does not hold true for the variable width. We define

$$W^*(\mathcal{C}) = \max\{W(G,H) : G \in \mathcal{C},\ H \notin \mathcal{C}\}.$$

Equivalently, $W^*(\mathcal{C})$ is equal to the minimum k such that $\mathcal{C}$ is definable in the infinitary logic $L^k_{\infty\omega}$; see, e.g., [19, Chapter 11]. Obviously, $W^*(\mathcal{C}) \le W(\mathcal{C})$, and we will see in Section 4.2 that this inequality can be strict. Summarizing, we have

$$W^*(\mathcal{C}) \le W(\mathcal{C}) \le D(\mathcal{C}). \tag{1}$$

The value of $W(\mathcal{C})$ admits the following characterization. If $W(G,H) \le k$, let $D^k(G,H)$ denote the minimum quantifier depth of a first-order k-variable sentence distinguishing G and H. We set $D^k(G,H) = \infty$ if $W(G,H) > k$. The following fact can be proved similarly to Lemma 1.

▶ **Lemma 2.** *$W(\mathcal{C}) > k$ if and only if there is a sequence of graph pairs (G_i, H_i) with $G_i \in \mathcal{C}$ and $H_i \notin \mathcal{C}$ such that $D^k(G_i, H_i) \to \infty$ as $i \to \infty$.*

Lemmas 1 and 2 reduce estimating the logical depth and width to estimating the parameters $D(G,H)$ and $D^k(G,H)$ over $G \in \mathcal{C}$ and $H \notin \mathcal{C}$. The first inequality in (1) can be used for obtaining lower bounds for $W(\mathcal{C})$ by estimating $W(G,H)$ over $G \in \mathcal{C}$ and $H \notin \mathcal{C}$. The parameters $D(G,H)$, $D^k(G,H)$, and $W(G,H)$ have a very useful combinatorial characterization.

In the *k-pebble Ehrenfeucht-Fraïssé game*, the board consists of two vertex-disjoint graphs G and H. Two players, *Spoiler* and *Duplicator* (or *he* and *she*) have equal sets of k pairwise different pebbles. In each round, Spoiler takes a pebble and puts it on a vertex in G or in H; then Duplicator has to put her copy of this pebble on a vertex of the other graph. Note that the pebbles can be reused and change their positions during the play. Duplicator's objective is to ensure that the pebbling determines a partial isomorphism between G and H after each round; when she fails, she immediately loses. The proof of the following facts can be found in [13]:

1. $D^k(G,H)$ is equal to the minimum d such that Spoiler has a winning strategy in the d-round k-pebble game on G and H.
2. $D(G,H)$ is equal to the minimum d such that Spoiler has a winning strategy in the d-round d-pebble game on G and H.
3. $W(G,H)$ is equal to the minimum k such that, for some d, Spoiler has a winning strategy in the d-round k-pebble game on G and H.

We are interested in the property of containing a specified induced subgraph. We write $F \sqsubset G$ to say that G contains an induced subgraph isomorphic to F. Thus, $\mathcal{I}(F) = \{G : F \sqsubset G\}$. Let $D[F] = D(\mathcal{I}(F))$ and, similarly, $W[F] = W(\mathcal{I}(F))$ and $W^*[F] = W^*(\mathcal{I}(F))$. Thus, $W^*[F]$ is the maximum $W(G,H)$ over all G containing an induced copy of F and all H not containing such a copy. As a particular case of (1), we have

$$W^*[F] \le W[F] \le D[F] \le \ell$$

for every F with ℓ vertices.

The vertex set of a graph G will be denoted by $V(G)$. Throughout the paper, we consider simple undirected graphs without loops. Let $\overline{G}$ denote the complement of G, that is, $V(\overline{G}) = V(G)$ and two vertices are adjacent in $\overline{G}$ exactly when they are not adjacent in G.

▶ **Lemma 3.** *$D[F] = D[\overline{F}]$, $W^*[F] = W^*[\overline{F}]$, and $W[F] = W[\overline{F}]$.*

Proof. The first equality follows from the equality $D(G,H) = D(\overline{G},\overline{H})$ by Lemma 1. Indeed, $F \sqsubset G$ iff $\overline{F} \sqsubset \overline{G}$. Therefore,

$$D[F] = \max\{\, D(G,H) : G \sqsupset F,\ H \not\sqsupset F\} = \max\left\{\, D(\overline{G},\overline{H}) : \overline{G} \sqsupset \overline{F},\ \overline{H} \not\sqsupset \overline{F}\right\}$$
$$= \max\left\{\, D(G,H) : G \sqsupset \overline{F},\ H \not\sqsupset \overline{F}\right\} = D[\overline{F}].$$

The second equality follows similarly from the equality $W(G,H) = W(\overline{G},\overline{H})$ by the definition of $W^*[F]$. The third equality follows from the equality $D^k(G,H) = D^k(\overline{G},\overline{H})$ by Lemma 2. ◀

2.1 Further graph-theoretic definitions

A graph G is called F-free if $F \not\sqsubset G$. The vertex-disjoint union of graphs G and H will be denoted by $G + H$. Correspondingly, sG is the vertex-disjoint union of s copies of G. The *lexicographic product* $A{\cdot}B$ of two graphs A and B is defined as follows: $V(A{\cdot}B) = V(A){\times}V(B)$, and (u,v) and (x,y) are adjacent in $A \cdot B$ if u and x are adjacent in A or if $u = x$ and v and y are adjacent in B. In other words, $A \cdot B$ is obtained from A by substituting each vertex u with an induced copy B_u of B and drawing all edges between B_u and B_x whenever u and x are adjacent.

A vertex is *isolated* if it has no adjacent vertex and *universal* if it is adjacent to all other vertices in the graph. Two vertices are called *twins* if they have the same adjacency to the rest of the graph.

Throughout the paper, $\log n$ means the logarithm base 2.

3 The extension index and a lower bound for $W^*[F]$

Let $k \geq 2$. By the *k-extension property* we mean the first-order sentence EA_k of quantifier depth k (also called the *kth extension axiom*) saying that, for every two disjoint sets $X, Y \subset V(G)$ with $|X \cup Y| < k$, there is a vertex $z \notin X \cup Y$ adjacent to all $x \in X$ and non-adjacent to all $y \in Y$. Note that EA_2 says exactly that a graph has neither isolated nor universal vertex. For convenience, we also set $EA_1 \stackrel{\text{def}}{=} \exists z(z = z)$.

Note that, if $G \models EA_k$ and F has at most k vertices, then $F \sqsubset G$. Suppose that F has more than 1 vertex. We define the *extension index* of F, denoted by $E[F]$, as the minimum k such that $H \models EA_k$ implies $F \sqsubset H$. Equivalently, $E[F]$ is the maximum k for which there is a graph H such that $H \models EA_{k-1}$ while $F \not\sqsubset H$. Note that $E[F] \leq \ell$ for any ℓ-vertex graph F.

▶ **Lemma 4.** $W^*[F] \geq E[F]$.

Proof. As easily seen, if both G and H have the $(k-1)$-extension property, then Duplicator has a winning strategy in the $(k-1)$-pebble Ehrenfeucht-Fraïssé game on graphs G and H and, hence, $W(G,H) \geq k$. Therefore, it suffices to show that there are G and H such that $F \sqsubset G$, $F \not\sqsubset H$, and both of them satisfy EA_{k-1} for $k = E[F]$. Such a graph H exists by the definition of the extension index. Such a graph G exists because, as very well known (see, e.g., [25]), for fixed k and ℓ a random graph $G(n,1/2)$ has the k-extension property and contains every ℓ-vertex graph as an induced subgraph with probability approaching 1 as n increases. Recall that $G(n,p)$ refers to the probability distribution on graphs with vertex set $\{1,\ldots,n\}$ where each two vertices are adjacent with probability p independently of the other pairs. ◀

▶ **Example 5.**

1. $E[P_3] = 3$ because $H = 2K_2$ is P_3-free and satisfies EA_2. By Lemma 4, $W^*[P_3] = W[P_3] = 3$.
2. $E[K_3] = 3$, as also certified by $H = 2K_2$ (or by $H = C_4$).

We can determine $E[K_\ell]$ for any ℓ using a relationship between $E[F]$ and the chromatic number of F.

▶ **Theorem 6.** *$E[F] \geq \chi(F)$.*

Proof. Let $k = \chi(F) - 1$. We have to show that there is a graph G having the kth extension property and containing no induced copy of F.

Let $T_{k,n}$ denote the k-partite Turán graph with kn vertices. The vertex set of $T_{k,n}$ is split into k vertex classes $V_1, \ldots, V_k$, each consisting of n vertices. Two vertices of $T_{k,n}$ are adjacent if and only if they belong to different vertex classes. Obviously, $\chi(T_{k,n}) = k$. Since $\chi(T_{k,n}) < \chi(F)$, the graph $T_{k,n}$ itself and any of its subgraphs do not contain an induced copy of F. Let $\mathbb{T}_{k,n}$ be a random subgraph of $T_{k,n}$, obtained from $T_{k,n}$ by deleting each edge with probability $1/2$, independently of the other edges. In order to prove the theorem, it suffices to show that $\mathbb{T}_{k,n}$ has the kth extension property with nonzero probability if n is chosen sufficiently large.

Consider two disjoint vertex sets X, Y in $\mathbb{T}_{k,n}$ such that $|X \cup Y| = k - 1$ and estimate the probability that they violate EA_k. Fix a vertex class V_m disjoint with $X \cup Y$. A particular vertex $z \in V_m$ is adjacent to all $x \in X$ and non-adjacent to all $y \in Y$ with probability 2^{-k+1}, and the converse happens with probability $1 - 2^{-k+1}$. The probability that none of the vertices in V_m has the "right" adjacency pattern to X and Y is equal to $(1 - 2^{-k+1})^n$. Using the inequality

$$1 - x \leq \mathrm{e}^{-x} \text{ for all reals } x, \tag{2}$$

we conclude that two sets X, Y violating EA_k exist with probability at most

$$\binom{kn}{k-1} 2^{k-1}(1-2^{-k+1})^n \leq \left(\frac{kn\,\mathrm{e}}{k-1}\right)^{k-1} 2^{k-1}\mathrm{e}^{-2^{-k+1}n} = \left(\frac{2k\,\mathrm{e}}{k-1}\right)^{k-1} \mathrm{e}^{(k-1)\ln n - 2^{-k+1}n},$$

which approaches 0 as n increases (since k is fixed). It follows that EA_k is violated by $\mathbb{T}_{k,n}$ with probability strictly less than 1 if n is chosen sufficiently large. ◀

▶ **Corollary 7.** *$E[K_\ell] = \ell$.*

It may seem plausible at first glance that $E[F] = \ell$ for every F with ℓ vertices. Nevertheless, in Section 4 we will see that this is not always the case as, for example, $E[F] = 3$ for F being the paw and the path on 4 vertices. The best general lower bound for $E[F]$ we can show is given by the following lemma.

▶ **Lemma 8.** *Let F be a graph with $\ell \geq 2$ vertices. Then*

$$E[F] \geq \lfloor \frac{1}{2}\,\ell - 2\log_2 \ell + 3 \rfloor.$$

Proof. The lemma is trivially true if $\ell \leq 15$ because in this case it just states that $E[F] \geq 2$. We, therefore, suppose that $\ell \geq 16$.

Denote $k = \lfloor \frac{1}{2}\,\ell - 2\log \ell + 2 \rfloor$. Suppose that ℓ is even and set $n = 2^{\ell/2-1}$. It suffices to show that the random graph $G(n, 1/2)$ with a non-zero probability has the k-extension property and simultaneously contains no induced copy of F.

The probability of $F \sqsubset G(n,1/2)$ is bounded from above by the value of

$$p(\ell,n) = n(n-1)(n-2)\cdots(n-\ell+1)2^{-\ell(\ell-1)/2}.$$

Since

$$2^{-\ell(\ell-1)/2} = (2n)^{-\ell+1} = 2^{-\ell+1}n^{-\ell+1} = \frac{1}{2}\,n^{-\ell-1},$$

we have

$$p(\ell,n) = \frac{n(n-1)(n-2)\cdots(n-\ell+1)}{2\,n^{\ell+1}} < \frac{1}{2n}.$$

It remains to prove that $G(n,1/2)$ has the k-extension property with probability at least $1/(2n)$.

The probability of $G(n,1/2) \not\models EA_k$ is bounded from above by the value of

$$q(n,k) = \binom{n}{k-1} 2^{k-1}(1-2^{-k+1})^{n-k+1}.$$

In its turn,

$$q(n,k) < 4\,n^{k-1}(1-2^{-k+1})^n.$$

By (2), the last value is bounded from above by

$$q'(n,k) = 4\,\exp\left(\ln n(k-1) - n\,2^{-k+1}\right).$$

Denote $k' = \frac{1}{2}\,\ell - 2\log\ell + 2$. Since the function $f(x) = \ln n\,x - n\,2^{-x}$ is monotonically increasing,

$$q(n,k) < q'(n,k') = 4\,n^{k'-1}\exp\left(-\frac{n}{2^{\ell/2-2\log\ell+1}}\right) = 4\,n^{k'-1}\exp\left(-\frac{\ell^2}{4}\right)$$

$$= 4\,n^{k'-1}(2n)^{-\log\mathrm{e}\,\ell/2} = 4\,n^{k'-1}2^{-\log\mathrm{e}\,\ell/2}n^{-\log\mathrm{e}\,\ell/2} = 4\,n^{k'-1}(2n)^{-\log\mathrm{e}}n^{-\log\mathrm{e}\,\ell/2}$$

$$= 4\,\mathrm{e}^{-1}n^{-\ell(\log\mathrm{e}-1)/2-2\log\ell-\log\mathrm{e}+1}.$$

For $\ell \geq 16$, this gives us $q(n,k) < 2\,n^{-11}$. Therefore, $G(n,1/2)$ has the k-extension property with probability more than $1-2\,n^{-11}$. This is well more than $1/(2n)$, as desired.

If ℓ is odd, set $n = 2^{(\ell-3)/2}$ and proceed similarly to above. ◀

▶ Remark. The bound of Lemma 8 cannot be much improved as long as the argument is based on $G(n,1/2)$. Indeed, it is known [15] that there is a function $\ell_0(n) = 2\log n - 2\log\log n + \Theta(1)$ such that the clique number of $G(n,1/2)$ is equal to $\ell_0(n)$ or to $\ell_0(n)+1$ with probability $1-o(1)$. In [16] it is shown that there is a function $k(n) = \log n - 2\log\log n + \Theta(1)$ such that, with probability $1-o(1)$, $G(n,1/2)$ satisfies $EA_{k(n)}$ but does not satisfy $EA_{k(n)+6}$. It follows that, if n is chosen so that $G(n,1/2)$ does not contain a subgraph K_ℓ with high probability, then $G(n,1/2)$ satisfies EA_k with non-negligible probability for, at best, $k = \frac{1}{2}\ell - \log\ell + \Theta(1)$.

As an immediate consequence of Lemmas 4 and 8 we obtain the following result.

▶ **Theorem 9.** *Let F be a graph with $\ell \geq 2$ vertices. Then*

$$W^*[F] > \frac{1}{2}\,\ell - 2\log_2\ell + 2.$$

4 Four-vertex subgraphs

Our next goal is to determine the values of $D[F]$, $W[F]$, and $W^*[F]$ for all graphs F with at most 4 vertices. It is enough to consider connected F, as follows from Lemma 3 and the fact that the complement of a disconnected graph is connected. The two connected 3-vertex graphs are considered in Example 5, and we now focus on connected graphs with 4 vertices. Recall that $W^*[K_4] = 4$ by Corollary 7 (or just because $W(K_4, K_3) = 4$).

4.1 The paw subgraph ($K_3 + e$)

▶ **Lemma 10** (Olariu [21]). *A graph H is paw-free if and only if each connected component of H is triangle-free or complete multipartite.*

Note that a graph B is complete multipartite iff the complement of B is a vertex-disjoint union of complete graphs. The latter condition means exactly that $\overline{B}$ is P_3-free. Thus, B is complete multipartite iff it is $(K_2 + K_1)$-free, where $K_2 + K_1 = \overline{P_3}$.

▶ **Theorem 11.** $D[K_3 + e] = W[K_3 + e] = W^*[K_3 + e] = E[K_3 + e] = 3$.

Proof. We have $E[K_3 + e] > 2$ because, for example, C_4 satisfies the 2nd extension axiom and does not contain $K_3 + e$ as a subgraph.

In order to prove that $D[K_3 + e] \leq 3$, we have to describe a winning strategy for Spoiler in the 3-round Ehrenfeucht-Fraïssé game on graphs $G \supset K_3 + e$ and $H \not\supset K_3 + e$. Let v_1, v_2, v_3, v_4 be vertices spanning a paw in G. We suppose that v_1 and v_2 have degree 2, v_3 has degree 3, and v_4 has degree 1 in this subgraph. In the first round Spoiler pebbles v_1. Suppose that Duplicator responds with a vertex u_1 in a connected component B of H. By Lemma 10, B is either K_3-free or a multipartite graph with at least three parts. In the former case Spoiler wins by pebbling v_2 and v_3. In the latter case Spoiler pebbles v_4 in the second round. The distance between v_1 and v_4 in G is 2. If Duplicator responds in a connected component of H other than B, then he loses in the next round. Therefore, Duplicator is forced in the second round to pebble a vertex u_2 in the same part of B that contains u_1. In this case, Spoiler wins by pebbling the vertex v_2. Indeed, this vertex is adjacent to v_1 and not adjacent to v_4, while u_1 and u_2 have the same adjacency to any other vertex in H. ◀

4.2 The path subgraph (P_4)

$F = P_4$ is a remarkable example showing that the parameters $W^*[F]$ and $W[F]$ can have different values. Specifically, we prove that $W^*[P_4] = 3$ and $W[P_4] = 4$.

▶ **Theorem 12.** $W^*[P_4] = E[P_4] = 3$.

The proof of Theorem 12 is based on a well-known characterization of the class of P_4-free graphs. A graph is called a *cograph* if it can be built from copies of the single-vertex graph K_1 by using disjoint unions and complementations. It is known [6] that a graph is P_4-free if and only if it is a cograph. For the proof of Theorem 12 we need the following definitions, that we borrow from [22].

We call G *complement-connected* if both G and $\overline{G}$ are connected. An inclusion-maximal complement-connected induced subgraph of G will be called a *complement-connected component* of G or, for brevity, a *cocomponent* of G. Cocomponents have no common vertices and their vertex sets form a partition of $V(G)$. Note that G is a cograph if and only if all cocomponents of G are single-vertex graphs.

The *decomposition* of G, denoted by $Dec\,G$, is the set of all connected components of G. Furthermore, given $i \geq 0$, we define the *depth i decomposition* $Dec_i\,G$ of G by

$$Dec_0\,G = Dec\,G \quad \text{and} \quad Dec_{i+1}\,G = \bigcup_{E \in Dec_i\,G} Dec\,\overline{E}.$$

Note that

$$\Pi_i = \{\, V(E) : E \in Dec_i\,G \} \tag{3}$$

is a partition of $V(G)$, and Π_{i+1} refines Π_i. Once the partition stabilizes, that is, $\Pi_{i+1} = \Pi_i$, it coincides with the partition of G into its cocomponents. The *depth i environment* of a vertex $v \in V(G)$, denoted by $Env_i(v)$, is the graph E in $Dec_i\,G$ containing v.

▶ **Lemma 13.** *Suppose that a graph G contains an induced copy of P_4 and let $U \subseteq V(G)$ be such that $G[U] \cong P_4$. Consider the 3-pebble Ehrenfeucht-Fraïssé game on G and another graph H. Let $x, x' \in V(G)$ and $y, y' \in V(H)$, and assume that the pairs x, y and x', y' were selected by the players in the same rounds. If $x, x' \in U$ and $Env_l(y) \neq Env_l(y')$, then Spoiler has a strategy allowing him to win in this position playing all the time in U and making no more than $2l + 2$ moves.*

Proof. We use induction on l. In the base case of $l = 0$, the vertices y and y' lie in different connected components of H, while the distance between x and x' in G is at most 3. Therefore, Spoiler is able to win with one extra pebble in 2 moves.

Let $l \geq 1$. Suppose that $Env_{l-1}(y) = Env_{l-1}(y') = E$ (for else Spoiler wins by the induction assumption). Note that $Env_l(y)$ and $Env_l(y')$ are connected components of $\overline{E}$. Since P_4 is self-complementary, $\overline{G[U]} \cong P_4$. Therefore, if Duplicator moves only in $V(E)$, Spoiler will win in at most 2 next moves. Once Duplicator makes one of these moves outside $V(E)$, this creates a position with two vertices $\tilde{x}$ and $\tilde{x}'$ pebbled by Spoiler in U such that $Env_{l-1}(\tilde{y}) \neq Env_{l-1}(\tilde{y}')$ for the corresponding vertices $\tilde{y}$ and $\tilde{y}'$ pebbled by Duplicator in H. The induction assumption applies. ◀

Proof of Theorem 12. Consider graphs $G \supset P_4$ and $H \not\supset P_4$. Since H is a cograph, $Dec_i\,H$ for some i consists of single-vertex graphs. By Lemma 13, this readily implies that $W(G, H) \leq 3$. Indeed, when Spoiler pebbles two vertices on an induced P_4 in G, then whatever Duplicator responds, this creates a position as in Lemma 13. Thus, $W^*[P_4] \leq 3$.

The lower bound $E[P_4] > 2$ is certified, for example, by C_4. ◀

Theorem 12 implies that the class of graphs containing an induced P_4 is definable in the infinitary logic $L^3_{\infty\omega}$. It turns out that this class is not definable in 3-variable first-order logic.

▶ **Theorem 14.** $W[P_4] = 4$.

Our proof of Theorem 14 is based on Lemma 2. It suffices to exhibit a sequence of graph pairs G_i, H_i such that G_i contains an induced copy of P_4, H_i does not, and $D^3(G_i, H_i) \to \infty$ as i increases.

Given a graph K, we define its ith power K^i by $K^1 = K$ and $K^{i+1} = \overline{K^i + K^i}$. Now, let $H_i = (K_1)^i$. This is a cograph and, therefore, $P_4 \not\subset H_i$ (which is also easy to see directly, using induction and the fact that P_4 is self-complementary).

In order to construct G_i, we use the lexicographic product of graphs; see Section 2. Fix a graph A satisfying the 3rd extension axiom and containing P_4 as an induced subgraph (a

large enough random graph has both of these properties with high probability). Now, let $G_i = H_i \cdot (A \cdot H_i)$. Obviously, $P_4 \sqsubset G_i$. Theorem 14 immediately follows by Lemma 2 from the following estimate.

▶ **Lemma 15.** $D^3(G_{4m+2}, H_{4m+2}) > m + 1$.

The proof of Lemma 15 can be found in the full version of the paper [27].

4.3 The claw ($K_{1,3}$) and the diamond ($K_4 \setminus e$) subgraphs

A *strongly regular graph* with parameters (n, k, λ, μ) is a regular graph with n vertices of degree k such that every two adjacent vertices have λ common neighbors and every two non-adjacent vertices have μ common neighbors. The simplest examples are sK_t (the vertex-disjoint union of s copies of the complete graph K_t) and their complements (complete s-partite graphs with each vertex class of size t). We call such strongly regular graphs *trivial*. A strongly regular graph is non-trivial exactly if $0 < \mu < k < n - 1$.

An example of a non-trivial strongly regular graph, that will be useful for us below, is the $m \times m$*-rook graph*. The vertex set of this graph is $\{\,(a, b) : 1 \le a, b \le m\}$, and two vertices (a_1, b_1) and (a_2, b_2) are adjacent if $a_1 = a_2$ or $b_1 = b_2$. In other words, each vertex represents a square of the $m \times m$ chess board, and two squares are adjacent if one is reachable from the other by a move of the rook. The $m \times m$-rook graph is strongly regular with parameters $(m^2, 2m - 2, m - 2, 2)$.

The condition $\lambda = 0$ means that a strongly regular graph is K_3-free. Every complete bipartite graph $K_{n,n} = \overline{2K_n}$ has this property and seven other triangle-free non-trivial graphs are known. It is open whether there is yet another such graph [12].

Suppose that H is a non-trivial non-triangle-free strongly regular graph with parameters (n, k, λ, μ). Thus, $\mu < k$ and it is also not hard to see that $\lambda < k - 1$ (otherwise every two adjacent vertices were twins and, by connectedness, every two vertices would be adjacent twins, implying that H is complete). These two inequalities readily imply that H satisfies the 3rd extension axiom.

▶ **Theorem 16.** $W[K_{1,3}] = W^*[K_{1,3}] = E[K_{1,3}] = 4$ *and* $W[K_4 \setminus e] = W^*[K_4 \setminus e] = E[K_4 \setminus e] = 4$.

Proof. We have to show that $E[K_{1,3}] > 3$ and $E[K_4 \setminus e] > 3$. By the discussion above, it suffices to exhibit a non-trivial non-triangle-free strongly regular graph that does not contain the claw and the diamond graphs as induced subgraphs. The 3×3-rook graph suits these needs. ◀

4.4 The cycle subgraph (C_4)

Let G be a connected graph. Given $u, v \in V(G)$, let $f^G_{i,j}(u, v)$ denote the number of vertices at distance i from u and at distance j from v. The graph G is called *distance-regular* if the number $f^G_{i,j}(u, v) = f^G_{i,j}(d)$ depends only on i, j, and the distance $d = d(u, v)$ between u and v. Note that such a graph is regular of degree $f^G_{1,1}(0)$. We call two distance-regular graphs G and H *similar* if

$$f^G_{i,j}(d) = 0 \iff f^H_{i,j}(d) = 0. \tag{4}$$

▶ **Lemma 17.** *If G and H are similar distance-regular graphs, then $W(G, H) > 3$.*

Proof. We show a strategy allowing Duplicator to win the 3-pebble game on G and H. In the first round she responds Spoiler's move arbitrarily. Let x and x' be the vertices pebbled in G and H respectively. Suppose that in the second round Spoiler pebbles a vertex y in G (the case that Spoiler plays in H is similar). Duplicator responds with a vertex y' in H such that $d(x', y') = d(x, y)$, which guarantees that she does not lose in this round. Such a choice of y' is possible because (4) implies that $f^H_{i,i}(0) > 0$ for $i = d(x, y)$.

For any subsequent round, assume that $x, y \in V(G)$ and $x', y' \in V(H)$ are occupied by two pairs of pebbles and that $d(x', y') = d(x, y)$. Suppose that Spoiler puts the third pebble on a vertex z in G (the case that Spoiler plays in H is similar). It is enough to notice that Duplicator can pebble a vertex z' in H such that $d(z', x') = d(z, x)$ and $d(z', y') = d(z, y)$. Such a vertex exists because (4) implies that $f^H_{i,j}(d) > 0$ for $i = d(z, x)$, $j = d(z, y)$, and $d = d(x, y)$. ◀

▶ **Theorem 18.** $W[C_4] = W^*[C_4] = 4$.

Proof. By Lemma 17, it suffices to exhibit similar distance-regular graphs G and H such that G contains an induced copy of C_4 and H does not. We can take G to be the cubic graph (the skeleton of the 3-dimensional cube) and $H = C_6$. ◀

5 Lower bounds over highly connected graphs

As we discussed in Section 1, in the case of a connected pattern graph F it is algorithmically motivated to consider the parameter $W_\kappa[F]$, which is the minimum variable width of a sentence defining the graph class $\mathcal{I}(F)$ correctly only over graphs of *sufficiently large* connectedness. More precisely, $W_\kappa[F]$ is equal to the minimum k for which there is a k-variable sentence Φ and a number s such that $G \models \Phi$ iff $F \sqsubset G$ for all s-connected graphs G. Moreover, we define

$$W^*_\kappa[F] = \min_s \max \{ W(G, H) : F \sqsubset G,\ F \not\sqsubset H, \text{ and both } G \text{ and } H \text{ are } s\text{-connected} \}.$$

This parameter is an analog of $W_\kappa[F]$ for the infinitary logic, and we have

$$W^*_\kappa[F] \leq W_\kappa[F] \leq W[F] \text{ and } W^*_\kappa[F] \leq W^*[F] \leq W[F].$$

The *join* of graphs A and B is denoted by $A * B$. Recall that this is the graph obtained from the disjoint union of A and B by adding all possible edges between a vertex of A and a vertex of B.

▶ **Lemma 19.** $W(A * B, A' * B) \geq W(A, A')$.

Proof. In the game on $A * B$ and $A' * B$, Duplicator can use her strategy for the game on A and A'. Each time that Spoiler moves in the B part of one graph, Duplicator just mirrors his move in the B part of the other graph. ◀

▶ **Lemma 20.** *Let F_0 be obtained from F by removing all universal vertices from this graph. Then $W^*_\kappa[F] \geq W^*[F_0]$.*

Proof. Let m denote the number of universal vertices in F. Suppose that $W^*[F_0] = W(G, H)$, where G contains an induced copy of F_0 and H does not. Let G' be obtained from G by adding new $s > m$ universal vertices, and let H' be defined similarly, i.e, $G' = G * K_s$ and $H' = H * K_s$. Note that G' contains an induced copy of F and H' still does not contain even an induced copy of F_0 (no new vertex of H' can appear in an induced copy of F_0 because it would be universal there). We have $W(G', H') \geq W(G, H)$ by Lemma 19. This proves the claim because G' and H' are s-connected and s can be chosen arbitrarily large. ◀

▶ **Theorem 21.**
1. $W^*_\kappa[F] = W^*[F]$ *whenever* F *has no universal vertex.*
2. $W^*_\kappa[F] = W^*[F]$ *whenever* $W^*[F] > 3$ *and* F *has no adjacent twins or no non-adjacent twins.*

Proof. Part 1 is an immediate consequence of Lemma 20. To establish Part 2, we have to prove that $W^*_\kappa[F] \geq W^*[F]$. Since with 3 pebbles Spoiler can force playing on connected components, the assumption $W^*[F] > 3$ implies that $W^*[F] = W(G, H)$ for some connected G and H such that $F \sqsubset G$ and $F \not\sqsubset H$. Assume that F has no adjacent twins. Consider $G' = G \cdot K_s$ and $H' = H \cdot K_s$ (recall that $\cdot$ denotes the lexicographic product of graphs). Note that G' and H' are s-connected and observe that $W(G', H') \geq W(G, H)$. Moreover, G' still contains an induced copy of F, and H' still does not (because if an induced subgraph of H' contains two vertices from the same K_s-part, they are adjacent twins in this subgraph). Since s can be chosen arbitrarily large, this implies that $W^*_\kappa[F] \geq W(G, H) = W^*[F]$, as required. If F has no non-adjacent twins, the same argument works with $G' = G \cdot \overline{K_s}$ and $H' = H \cdot \overline{K_s}$. ◀

An example of a graph to which Theorem 21 is non-applicable is the diamond. It has universal vertices and both adjacent and non-adjacent twins.

If an ℓ-vertex pattern graph F has no universal vertex, then $W^*_\kappa[F] = W^*[F]$ and, therefore, $W^*_\kappa[F] \geq (\frac{1}{2} - o(1))\ell$ by Theorem 9. Lemma 20 works well also for F with few universal vertices. For example, we have $W^*_\kappa[K_{1,\ell}] \geq W^*[\overline{K_\ell}] = W^*[K_\ell] = \ell$. However, if F has many universal vertices, then we need a different approach.

Similarly to $E[F]$, we define $E_\kappa[F]$ to be the maximum k such that, for each s, there is an s-connected graph H with $H \models EA_{k-1}$ while $F \not\sqsubset H$.

The following relations are easy to prove similarly to Lemma 4 and Theorem 6 (note that, for each fixed s, the random multipartite graph $\mathbb{T}_{k,n}$ in the proof of Theorem 6 is s-connected with high probability).

▶ **Lemma 22.** $W^*_\kappa[F] \geq E_\kappa[F] \geq \chi(F)$.

▶ **Theorem 23.** *If* F *has* ℓ *vertices, then* $W^*_\kappa[F] > \frac{1}{3}\ell - \frac{4}{3}\log_2 \ell$.

Proof. Denote the number of universal vertices in F by m, and let F_0 be obtained from F by removing all these vertices. By Lemma 20 and Theorem 9,

$$W^*_\kappa[F] \geq W^*[F_0] > \frac{1}{2}(\ell - m) - 2\log \ell. \tag{5}$$

By Lemma 22,

$$W^*_\kappa[F] \geq \chi(F) \geq m. \tag{6}$$

Combining the bounds (5) and (6), we obtain

$$3\, W^*_\kappa[F] > \ell - 4\log \ell,$$

which implies the bound stated in the theorem. ◀

Finally, we determine the values of $W^*_\kappa[F]$ and $W_\kappa[F]$ for small connected pattern graphs. As a particular case of Lemma 22, we have $W^*_\kappa[K_3] = 3$ and $W^*_\kappa[K_4] = 4$. According to the discussion in Section 1, we have $W_\kappa[P_3] \leq D_v[P_3] \leq 2$. On the other hand, $W^*_\kappa[P_3] \geq 2$ just because there are highly connected graphs with an induced copy of P_3 and without it, for example, $K_n \setminus e$ and K_n respectively.

▶ **Theorem 24.**

1. $W^*_\kappa[K_3 + e] = W[K_3 + e] = 3$;
2. $W^*_\kappa[P_4] = W^*[P_4] = 3$ *and* $W_\kappa[P_4] = W[P_4] = 4$;
3. $W^*_\kappa[F] = W[F] = 4$ *for all remaining connected* F *on 4 vertices.*

Proof.

1. In the trivial direction, $W^*_\kappa[K_3 + e] \leq W[K_3 + e] = 3$, the equality being established in Theorem 11. On the other hand, we have

$$W^*_\kappa[K_3 + e] \geq W^*[K_2 + K_1] = W^*[\overline{K_2 + K_1}] = W^*[P_3] \geq E[P_3] = 3,$$

   where we use Lemma 20, Lemma 3, and Part 1 of Example 5.
2. We have $W^*_\kappa[P_4] = W^*[P_4] = 3$ by Part 1 of Theorem 21 and by Theorem 12. Since Lemma 20 easily extends to the relation $W_\kappa[F] \geq W[F_0]$, we also have $W_\kappa[P_4] = W[P_4] = 4$.
3. We have $W^*_\kappa[C_4] = W^*[C_4] = 4$ by Part 1 of Theorem 21 and by Theorem 18.

We also have $W^*_\kappa[K_{1,3}] = E_\kappa[K_{1,3}] = 4$ and $W^*_\kappa[K_4 \setminus e] = E_\kappa[K_4 \setminus e] = 4$ by the first inequality in Lemma 22. Indeed, $E_\kappa[K_{1,3}] = E_\kappa[K_4 \setminus e] = 4$ because the $m \times m$ rook's graph is a strongly regular graph containing no induced copy of $K_{1,3}$ and no induced copy of $K_4 \setminus e$. Moreover, the $m \times m$ rook's graph is $(2m-2)$-connected by the following general fact: The connectivity of a connected strongly regular graph equals its vertex degree [2]. ◀

6 Conclusion

- The equality $D[K_3 + e] = 3$ is currently the only example we know that shows that $D[F]$ and $W[F]$ can be strictly less than the number of vertices in F. Are there other such graphs? Are there infinitely many of them? On the other hand, we only know that $W[F] \geq (\frac{1}{2} - o(1))\ell$ for all F with ℓ vertices. This does not even exclude the possibility that the time bound $O(n^{W[F]})$ for Induced Graph Isomorphism can be better than the Nešetřil-Poljak bound $O(n^{(\omega/3)\ell+c})$ for infinitely many pattern graphs F.
- An example of $F = P_4$ shows that $W^*[F]$ can be strictly less than $W[F]$ but we do not know how far apart from each other $W^*[F]$ and $W[F]$ can generally be.
- Note that Lemmas 4 and 8 show the following hierarchy of graph parameters:

$$(1/2 - o(1))\,\ell \leq E[F] \leq W^*[F] \leq W[F] \leq D[F] \leq \ell, \tag{7}$$

  where ℓ is the number of vertices in F. It seems that we currently have no example separating the parameters $W[F]$ and $D[F]$ or the parameters $E[F]$ and $W^*[F]$. An important question is whether or not $E[F] = (1 - o(1))\ell$.
- It follows from (7) that $W[F] \leq (2 + o(1))W^*[F]$. In other terms, in the context of Induced Subgraph Isomorphism, the infinitary logic cannot be much more succinct than the standard first-order logic with respect to the number of variables. More generally, is it true that $W(\mathcal{C}) = O(W^*(\mathcal{C}))$ for all first-order definable graph properties?
- We have checked that $D[F] = W[F] = \ell$ for all F with $\ell \leq 4$ vertices excepting for the paw graph and its complement. Since it remains open if the equality holds true for all larger graphs, it seems reasonable to examine the pattern graphs on 5 vertices. If there is a 5-vertex F with $W[F] = 4$, the resulting decision procedure for $\mathcal{I}(F)$ would be competitive to (or, at least comparable with) the currently known algorithmic results for 5-vertex induced subgraphs [9, 28].
- We have a (rather trivial) example of $F = P_3$ showing that $W_\kappa[F]$ can be strictly smaller than $W[F]$. Are there other such graphs?

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

2 Andries E. Brouwer and Dale M. Mesner. The connectivity of strongly regular graphs. *Eur. J. Comb.*, 6(3):215–216, 1985. doi:10.1016/S0195-6698(85)80030-5.

3 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

4 Yijia Chen and Jörg Flum. On parameterized path and chordless path problems. In *Proc. of the 22nd Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 250–263. IEEE Computer Society, 2007. doi:10.1109/CCC.2007.21.

5 Derek G. Corneil, Yehoshua Perl, and Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926–934, 1985. doi:10.1137/0214065.

6 D. G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Appl. Math.*, 3(3):163–174, 1981. doi:10.1016/0166-218X(81)90013-5.

7 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.

8 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

9 Peter Floderus, Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Detecting and counting small pattern graphs. *SIAM J. Discrete Math.*, 29(3):1322–1339, 2015. doi:10.1137/140978211.

10 Peter Floderus, Mirosław Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Induced subgraph isomorphism: Are some patterns substantially easier than others? *Theoretical Computer Science*, 605:119–128, 2015. doi:10.1016/j.tcs.2015.09.001.

11 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. of the Int. Symposium on Symbolic and Algebraic Computation (ISSAC'14)*, pages 296–303. ACM, 2014. doi:10.1145/2608628.2608664.

12 Chris D. Godsil. Problems in algebraic combinatorics. *Electr. J. Comb.*, 2:F#1, 1995. URL: http://www.combinatorics.org/Volume_2/PDFFiles/v2i1f1.pdf.

13 Neil Immerman. *Descriptive complexity*. New York, NY: Springer, 1999. doi:10.1007/978-1-4612-0539-5.

14 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. doi:10.1137/0207033.

15 Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random graphs*. New York, Berlin: Wiley, 2000.

16 Jeong Han Kim, Oleg Pikhurko, Joel H. Spencer, and Oleg Verbitsky. How complex are random graphs in first order logic? *Random Struct. Algorithms*, 26(1-2):119–145, 2005. doi:10.1002/rsa.20049.

17 Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Inf. Process. Lett.*, 74(3-4):115–121, 2000. doi:10.1016/S0020-0190(00)00047-8.

18 Yuan Li, Alexander A. Razborov, and Benjamin Rossman. On the AC^0 complexity of Subgraph Isomorphism. In *Proc. of the 55th IEEE Ann. Symposium on Foundations of Computer Science (FOCS'14)*, pages 344–353. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.44.

19 Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. doi:10.1007/978-3-662-07003-1.

20 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentat. Math. Univ. Carol.*, 26:415–419, 1985.

21 Stephan Olariu. Paw-free graphs. *Inf. Process. Lett.*, 28:53–54, 1988. doi:10.1016/0020-0190(88)90143-3.

22 Oleg Pikhurko, Joel Spencer, and Oleg Verbitsky. Decomposable graphs and definitions with no quantifier alternation. *Eur. J. Comb.*, 28(8):2264–2283, 2007. doi:10.1016/j.ejc.2007.04.016.

23 Oleg Pikhurko and Oleg Verbitsky. Logical complexity of graphs: a survey. In Martin Grohe and Janos Makowsky, editors, *Model theoretic methods in finite combinatorics*, volume 558 of *Contemporary Mathematics*, pages 129–179. American Mathematical Society (AMS), Providence, RI, 2011.

24 Benjamin Rossman. On the constant-depth complexity of k-clique. In *Proc. of the 40th Ann. ACM Symposium on Theory of Computing (STOC'08)*, pages 721–730. ACM, 2008. doi:10.1145/1374376.1374480.

25 Joel H. Spencer. *The strange logic of random graphs.* Berlin: Springer, 2001. doi:10.1007/978-3-662-04538-1.

26 Oleg Verbitsky and Maksim Zhukovskii. The descriptive complexity of Subgraph Isomorphism without numerics. In *Proc. of the 12th Int. Computer Science Symposium in Russia (CSR'17)*, volume 10304 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2017. A full version: http://arxiv.org/abs/1607.08067. doi:10.1007/978-3-319-58747-9_27.

27 Oleg Verbitsky and Maksim Zhukovskii. On the first-order complexity of Induced Subgraph Isomorphism. E-print, 2017. URL: http://arxiv.org/abs/1704.02237.

28 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In *Proc. of the 26th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 1671–1680. SIAM, 2015. doi:10.1137/1.9781611973730.111.

Strategies with Parallel Causes

Marc de Visme[1] and Glynn Winskel[2]

1 **École Normale Supérieure de Paris, Paris, France**
2 **Computer Laboratory, University of Cambridge, Cambridge, UK**

Abstract

We imagine a team Player engaging a team Opponent in a distributed game. Such games and their strategies have been formalised within event structures. However there are limitations in founding strategies on traditional event structures. Sometimes a probabilistic distributed strategy relies on benign races where, intuitively, several members of team Player may race each other to make a common move. Although there exist event structures which support such parallel causes, in which an event is enabled in several compatible ways, they do not support an operation of hiding central to the composition of strategies; nor do they support probability adequately. An extension of traditional event structures is devised which supports parallel causes and hiding, as well as the mix of probability and nondeterminism needed to account for probabilistic distributed strategies. The extension is located within existing models for concurrency and tested in the construction of a bicategory of probabilistic distributed strategies with parallel causes.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation

Keywords and phrases Games, Strategies, Event Structures, Parallel Causes, Probability

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.41

1 Introduction

This article addresses a fundamental, potentially widespread issue of which few are aware. It concerns the accurate modelling of parallel causes in probabilistic distributed strategies; we are thinking for instance of a strategy in which it is advantageous to allow two or more members of the same team to race each other cooperatively, without conflict, to perform some common move. It fixes the absence of a computational model which simultaneously handles parallel causes, probability and an operation of hiding internal events; it provides such a model, locates it via adjunctions within existing models and tests it in the construction of a bicategory of probabilistic distributed strategies supporting parallel causes.

Consider probabilistic distributed games between two teams, Player and Opponent. To set the scene, imagine a simple distributed game in which team Opponent can perform two moves, called 1 and 2, far apart from each other, and that team Player can just make one move, 3. Suppose that for Player to win they must make their move iff Opponent makes one or more of their moves. Informally Player can win by assigning two members of their team, one to watch out for the Opponent move 1 and the other Opponent move 2. When either watcher sees their respective Opponent move they run back and make the Player move 3. Opponent could possibly play both 1 and 2 in which case both watchers would run back and could make their move cooperatively together. Provided the watchers are perfectly reliable this provides a winning probabilistic strategy for Player. No matter how Opponent chooses to play or not play their moves, Player will win; if Opponent is completely inactive the watchers wait forever but then Player does win, eventually.

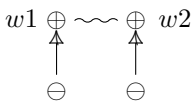
We can imagine variations in which the watchers are only reliable with certain probabilities, independent or correlated, with a consequent reduction in the probability of Player winning

against Opponent strategies. In such a probabilistic strategy Player can only determine probabilities of their moves conditionally on those of Opponent. Because Player has no say in the probabilities of Opponent moves beyond those determined by causal dependencies of the strategy we are led to a *Limited Markov Condition*, of the kind discussed in [6]:

> **(LMC)** In a configuration x in which both a Player move $\oplus$ and an Opponent move $\ominus$ could occur individually, if the Player move and the Opponent move are causally independent, then they are probabilistically independent; in a strategy for Player, $\mathrm{Prob}(\oplus \mid x, \ominus) = \mathrm{Prob}(\oplus \mid x)$.

Note we do not expect that in all strategies for Player that two causally independent Player moves are necessarily probabilistically independent; in fact, because composition of strategies involves hiding internal moves such a property would not generally be preserved by composition.

Let us try to describe the informal winning strategy above in terms of event structures. In 'prime' event structures in which causal dependency is expressed as a partial order, an event is causally dependent on a unique set of events, *viz.* those events below it in the partial order. For this reason within prime event structures we are forced to split the Player move 3 into two events one for each watcher making the move, one $w1$ dependent on Opponent move 1 and the other $w2$ on Opponent move 2. The two moves of the two watchers stand for the same Player move in the game. Because of this they are in conflict (or inconsistent) with each other.[1] We end up with the event structure drawn below:



The polarities $+$ and $-$ signify moves of Player and Opponent, respectively. The arrows represent the (immediate) causal dependencies and the wiggly line conflict. As far as purely nondeterministic behaviour goes, we have expressed the informal strategy well: no matter how Opponent makes or doesn't make their moves any maximal play of Player is assured to win. However consider assigning conditional probabilities to the watcher moves. Suppose the probability of $w1$ conditional on Opponent event 1 is p_1, *i.e.* $\mathrm{Prob}(w1 \mid 1) = \mathrm{Prob}(w1, 1 \mid 1) = p_1$ and that similarly for $w2$ its conditional probability $\mathrm{Prob}(w2 \mid 2) = p_2$. Given that move $w1$ of Player and move 2 of Opponent are causally independent, from (LMC) we expect that $w1$ is probabilistically independent of move 2, *i.e.*whether Opponent chooses to make move 2 or not should have no influence on the watcher of move 1:

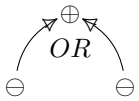
$$\mathrm{Prob}(w1 \mid 1, 2) = \mathrm{Prob}(w1 \mid 1) = p_1\,;\ \text{and similarly, } \mathrm{Prob}(w2 \mid 1, 2) = \mathrm{Prob}(w2 \mid 2) = p_2\,.$$

But $w1$ and $w2$ are in conflict, so mutually exclusive, and can each occur individually when 1 and 2 have occurred, ensuring that $p_1 + p_2 \leq 1$ – we haven't insisted on one or the other occurring, the reason why we have not written equality. The best Player can do is assign $p_1 = p_2 = 1/2$. Against a counter-strategy with Opponent playing one of their two moves with probability $1/2$ this strategy only wins half the time. We have clearly failed to express the informal winning strategy accurately!

Present notions of "concurrent strategies," the most general of which are presented in [11], are or can be expressed using prime event structures. If we are to be able to express the

[1] Technically, the conflict is forced by the nature of maps of event structures; a map reflects the atomicity of events and cannot send distinct consistent events to a common event.

intuitive strategy which wins with certainty we need to develop distributed probabilistic strategies which allow *parallel causes* in which an event can be enabled in distinct but compatible ways. 'General' event structures are one such model [10]. In the informal strategy described above both Opponent moves would individually enable the Player move, with all events being consistent, illustrated below:



But as we shall see general event structures do not support an appropriate operation of hiding central to the composition of strategies. Nor is it clear how within general event structures one could express the variant of the strategy above in which the two watchers succeed in reporting with different probabilities while respecting LMC – see Section 3.1.

It has been necessary to develop a new model – *event structures with disjunctive causes* (edc's) – which support hiding and probability adequately, and into which both prime and general event structures embed. Conceptually, one is forced to objectify cause in a way that is reminiscent of formal proof being an objectification of theoremhood. Formally, this is achieved by extending prime event structures with an equivalence relation; the equivalence classes are thought of as 'disjunctive events' of which the representatives are 'prime causes.' In this way causes may conflict or not, possess probabilities, and be correlated or independent. The new model provides a foundation on which to build a rich theory of probabilistic distributed strategies with parallel causes. Even without probability, it provides a new bicategory of *deterministic* parallel strategies, including *e.g.* deterministic strategies for "parallel or" and McCarthy's amb [4].

Full proofs can be found in [12]*, Chapter 16, 17. Appendix A summarises the simple instances of concepts we borrow from enriched categories* [2, 3] *and 2-categories* [7].

2 Event structures

We start with event structures. In their simplest form, that of 'prime' event structures, they occupy a central position in models for concurrent computation, both "interleaving" and "causal" [13], and can claim to be the concurrent or causal analogue of trees.

2.1 Prime event structures

A *(prime) event structure* comprises $(E, \leq, \mathrm{Con})$, consisting of a set E of *events* which are partially ordered by $\leq$, the *causal dependency relation*, and a nonempty *consistency relation* Con consisting of finite subsets of E. The relation $e' \leq e$ expresses that event e causally depends on the previous occurrence of event e'. That a finite subset of events is consistent conveys that its events can occur together by some stage in the evolution of the process. Together the relations satisfy several axioms:

$[e] =_{\mathrm{def}} \{e' \mid e' \leq e\}$ is finite for all $e \in E$,
$\{e\} \in \mathrm{Con}$ for all $e \in E$,
$Y \subseteq X \in \mathrm{Con}$ implies $Y \in \mathrm{Con}$, and
$X \in \mathrm{Con}$ & $e \leq e' \in X$ implies $X \cup \{e\} \in \mathrm{Con}$.

There is an accompanying notion of state, or history, those events that may occur up to some stage in the behaviour of the process described. A *configuration* is a, possibly infinite, set of

events $x \subseteq E$ which is: *consistent,* $X \subseteq x$ and X is finite implies $X \in \mathrm{Con}$; and *down-closed,* $e' \leq e \in x$ implies $e' \in x$.

Two events e, e' are considered to be causally independent, and called *concurrent* if the set $\{e, e'\}$ is in Con and neither event is causally dependent on the other. The relation of *immediate* dependency $e \rightarrowtriangle e'$ means e and e' are distinct with $e \leq e'$ and no event in between. We write $[X]$ for the down-closure of a subset of events X. Write $\mathcal{C}^\infty(E)$ for the configurations of E and $\mathcal{C}(E)$ for its finite configurations.

It will be very useful to relate event structures by maps. A *map* of event structures $f : E \to E'$ is a partial function f from E to E' such that the image of a configuration x is a configuration fx and any event of fx arises as the image of a unique event of x; the map is thus locally injective w.r.t. a configuration x. Maps compose as partial functions. Write $\mathcal{E}$ for the ensuing category.

A map $f : E \to E'$ reflects causal dependency locally, in the sense that if e, e' are events in a configuration x of E for which $f(e') \leq f(e)$ in E', then $e' \leq e$ also in E; the event structure E inherits causal dependencies from the event structure E' via the map f. Consequently, a map preserves concurrency: if two events are concurrent, then their images if defined are also concurrent. In general a map of event structures need not preserve causal dependency.

2.2 General event structures

In contrast, a *general event structure* [9, 10] permits an event to be caused disjunctively in several ways, possibly coexisting in parallel, *i.e.* parallel causes. A general event structure comprises $(E, \mathrm{Con}, \vdash)$ where E is a set of event occurrences, the consistency relation Con is a non-empty collection of finite subsets of E, and the *enabling relation* $\vdash$ is a relation in $\mathrm{Con} \times E$ such that

$$\begin{array}{l} X \subseteq Y \in \mathrm{Con} \implies X \in \mathrm{Con}\,, \text{ and} \\ Y \in \mathrm{Con} \;\&\; Y \supseteq X \;\&\; X \vdash e \implies Y \vdash e\,. \end{array}$$

A *configuration* is a subset of E which is: *consistent,* $X \subseteq_{\mathrm{fin}} x \implies X \in \mathrm{Con}$; and *secured,* $\forall e \in x,\ \exists e_1, \cdots, e_n \in x,\ \ e_n = e\ \ \&\ \forall i \leq n, \{e_1, \cdots, e_{i-1}\} \vdash\ e_i$. Again we write $\mathcal{C}^\infty(E)$ for the configurations of E and $\mathcal{C}(E)$ for its finite configurations.

The notion of an event e being enabled in a configuration has been expressed through the existence of a securing chain $e_1, \cdots, e_n$, with $e_n = e$, within the configuration. The securing chain represents a *complete enabling* of e in the sense that every event in the securing chain is itself enabled by earlier members of the chain. But just as mathematical proofs need not be sequences, so can one imagine more refined ways in which to express complete enablings. Later the idea that complete enablings can be more generally expressed as partial orders of events in which all events are enabled by earlier events in the order – "causal realisations" – will play an important role in unfolding general event structures to structures supporting hiding and parallel causes.

A *map* $f : (E, \mathrm{Con}, \vdash) \to (E', \mathrm{Con}', \vdash')$ of general event structures is a partial function $f : E \rightharpoonup E'$ such that

$$\begin{array}{r} X \in \mathrm{Con} \implies fX \in \mathrm{Con}'\,, \\ \forall e_1, e_2 \in X \in \mathrm{Con},\ f(e_1) = f(e_2) \implies e_1 = e_2\,, \text{ and} \\ X \vdash e \;\&\; f(e) \text{ is defined} \implies fX \vdash' f(e)\,. \end{array}$$

Maps compose as partial functions with identity maps being identity functions. Write $\mathcal{G}$ for the category of general event structures.

We can characterise those families of configurations arising from a general event structure. W.r.t. a family of subsets $\mathcal{F}$, a subset X of $\mathcal{F}$ is *compatible* (in $\mathcal{F}$), written $X\uparrow$, if there is $y \in \mathcal{F}$ such that $x \subseteq y$ for all $x \in X$; we write $x \uparrow y$ for $\{x, y\}\uparrow$. Say a subset is *finitely compatible* iff every finite subset is compatible.

A *family of configurations* comprises a family $\mathcal{F}$ of sets such that if $X \subseteq \mathcal{F}$ is finitely compatible in $\mathcal{F}$ then $\bigcup X \in F$; and if $e \in x \in \mathcal{F}$ there is a securing chain $e_1, \cdots, e_n = e$ in x such that $\{e_1, \cdots, e_i\} \in \mathcal{F}$ for all $i \leq n$. The elements of the underlying set $\bigcup \mathcal{F}$ are its *events*. Such a family is *stable* when for any compatible non-empty subset X of $\mathcal{F}$ its intersection $\bigcap X$ is a member of $\mathcal{F}$.

For configurations x, y, we use $x \mathrel{-\!\!\subset} y$ to mean y covers x, *i.e.* $x \subset y$ with nothing in between, and $x \overset{e}{\mathrel{-\!\!\subset}} y$ to mean $x \cup \{e\} = y$ for an event $e \notin x$. We sometimes use $x \overset{e}{\mathrel{-\!\!\subset}}$, expressing that event e is enabled at configuration x, when $x \overset{e}{\mathrel{-\!\!\subset}} y$ for some y.

A map between families of configurations from $\mathcal{A}$ to $\mathcal{B}$ is a partial function $f : \bigcup \mathcal{A} \rightharpoonup \bigcup \mathcal{B}$ between their events such that $fx \in \mathcal{B}$ if $x \in \mathcal{A}$ and any event of fx arises as the image of a unique event of x. Maps compose as partial functions.

The forgetful functor taking a general event structure to its family of configurations has a left adjoint, which constructs a canonical general event structure from a family: given $\mathcal{A}$, a family of configurations with underlying events A, construct a general event structure $(A, \mathrm{Con}, \vdash)$ with $X \in \mathrm{Con}$ iff $X \subseteq_{\mathrm{fin}} y$, for some $y \in \mathcal{A}$; and with $X \vdash a$ iff $a \in A$, $X \in \mathrm{Con}$ and $e \in y \subseteq X \cup \{a\}$, for some $y \in \mathcal{A}$.

The above yields a coreflection of families of configurations in general event structures. It cuts down to an equivalence between families of configurations and *replete* general event structures. A general event structure $(E, \mathrm{Con}, \vdash)$ is *replete* iff

$$\begin{aligned} \forall e \in E,\ \exists X \in \mathrm{Con},\ X \vdash e\,, \\ \forall X \in \mathrm{Con},\ \exists x \in \mathcal{C}(E),\ X \subseteq x \text{ and} \\ X \vdash e \implies \exists x \in \mathcal{C}(E),\ e \in x \ \&\ x \subseteq X \cup \{e\}\,. \end{aligned}$$

2.3 On relating prime and general event structures

Clearly a prime event structure $(P, \leq, \mathrm{Con})$ can be identified with a (replete) general event structure $(P, \vdash, \mathrm{Con})$ by taking $X \vdash p$ iff $X \in \mathrm{Con}$ & $[p] \subseteq X \cup \{p\}$. Indeed under this identification there is a full and faithful embedding of $\mathcal{E}$ in $\mathcal{G}$. However (contrary to the claim in [10]) there is no adjoint to this embedding. This leaves open the issue of providing a canonical way to describe a general event structure as a prime event structure. This issue has arisen as a central problem in reversible computation [1] and now more recently in the present limitation of concurrent strategies described in the introduction. A corollary of our work will be that the embedding of prime into general event structures does have a *pseudo* right adjoint which unfolds a general event structure to a prime event structure, got at the slight cost of enriching prime event structures with equivalence relations.

3 Problems with general event structures

Why not settle for general event structures as a foundation for distributed strategies? Because although they allow parallel causes, they don't generally support hiding, so composition of strategies; nor do they support probability generally enough.[2]

[2] Should we only be interested in deterministic, non-probabilistic strategies, general event structures do support pullback and hiding required in the composition of strategies [12]. Nondeterministic or probabilistic strategies with parallel causes require an extension such as ese's or edc's, defined shortly.

3.1 Probability and parallel causes

We return to the general-event-structure description of the strategy in the introduction. To turn this into a probabilistic strategy for Player we should assign probabilities to configurations conditional on Opponent moves. The watcher of Opponent move 1 is causally independent of Opponent move 2. Given this we might expect that the probability of the watcher of 1 making the Player move 3 should be probabilistically independent of move 2; after all, both moves 3 and 2 can occur concurrently from configuration $\{1\}$. Applying LMC naively would yield $\mathrm{Prob}(1,3 \mid 1) = \mathrm{Prob}(1,2,3 \mid 1,2)$. But similarly, $\mathrm{Prob}(2,3 \mid 2) = \mathrm{Prob}(1,2,3 \mid 1,2)$, which forces $\mathrm{Prob}(1,3 \mid 1) = \mathrm{Prob}(2,3 \mid 2)$, *i.e.* that the conditional probabilities of the two watchers succeeding are the same! In blurring the distinct ways in which move 3 can be caused we have obscured causal independence which has led us to identify possibly distinct probabilities.

3.2 Hiding

With one exception, all the operations used in building strategies and, in particular, the bicategory of concurrent strategies [8], extend to general event structures. The one exception, that of hiding, is crucial in ensuring composition of strategies yields a bicategory.

Consider a general event structure with *events* a, b, c, d and e; *enabling* (1) $b, c \vdash e$ and (2) $d \vdash e$, with all events other than e being enabled by the empty set; and *consistency* in which all subsets are consistent unless they contain the events a and b. Any configuration will satisfy the assertion $(a \wedge e) \implies d$ because if e has occurred it has to have been enabled by (1) or (2) and if a has occurred its conflict with b has prevented the enabling (1), so e can only have occurred via enabling (2).

Now imagine the event b is hidden, so allowed to occur invisibly in the background. The configurations after hiding are those obtained by hiding (*i.e.* removing) the invisible event b from the configurations of the original event structure. The assertion $(a \wedge e) \implies d$ will still hold of the configurations after hiding.

There isn't a general event structure with events a, c, d and e, and configurations those which result when we hide (remove) b from the configurations of the original event structure.[3]

Precisely the same problem can arise in the composition (with hiding) of nondeterministic strategies based on general event structures. To obtain a bicategory of strategies with disjunctive causes we need to support hiding. We need to look for structures more general than general event structures. The example above gives a clue: inconsistency should be lifted from an inconsistency between events to an inconsistency between enablings.

4 Adding disjunctive causes

To cope with disjunctive causes and hiding we must go beyond general event structures. We introduce structures in which we *objectify* cause; a minimal complete enabling is no longer an instance of a relation but a structure that realises that instance (*cf.* a judgement of theorem-hood in contrast to a proof).

[3] One way to see this is to observe that amongst the configurations after hiding we have $\{c\} \text{–}\subset \{c, e\}$ and $\{c\} \text{–}\subset \{a, c\}$ where both $\{c, e\}$ and $\{a, c\}$ have upper bound $\{a, c, d, e\}$, and yet $\{a, c, e\}$ is not a configuration after hiding as it fails to satisfy the assertion $(a \wedge e) \implies d$. In configurations of a general event structure if $x \text{–}\subset y$ and $x \text{–}\subset z$ and y and z are compatible, then $y \cup z$ is a configuration.

Fortunately we can do this while staying close to prime event structures. The twist is to regard "disjunctive events" as comprising subsets of events of a prime event structure, the events of which are now to be thought of as representing "prime causes" standing for minimal complete enablings. Technically, we do this by extending prime event structures with an equivalence relation on events.

In detail, an *event structure with equivalence* (an ese) is a structure $(P, \leq, \mathrm{Con}, \equiv)$ where $(P, \leq, \mathrm{Con})$ is a (prime) event structure and $\equiv$ is an equivalence relation on P.

An ese dissociates the two roles of enabling and atomic action conflated in the events of a prime event structures. The intention is that the events p of P, or really their corresponding down-closures $[p]$, describe minimal complete enablings, *prime causes*, while the $\equiv$-equivalence classes of P represent *disjunctive events*: p is a prime cause of the disjunctive event $\{p\}_{\equiv}$. Notice there may be several prime *disjunctive causes* of the same disjunctive event and that these may be *parallel causes* in the sense that they are consistent with each other and not related in the order $\leq$.

A *configuration* of the ese is a configuration of $(P, \leq, \mathrm{Con})$ and we shall use the notation of earlier on event structures $\mathcal{C}^{\infty}(P)$ and $\mathcal{C}(P)$ for its configurations, respectively finite configurations. We say a configuration is *unambiguous* if it has no two distinct elements which are $\equiv$-equivalent. We modify the relation of concurrency a little and say $p_1, p_2 \in P$ are *concurrent* and write $p_1 \mathrm{co}\, p_2$ iff $[p_1] \cup [p_2]$ is an *unambiguous* configuration of P and neither $p_1 \leq p_2$ nor $p_2 \leq p_1$.

When the equivalence relation $\equiv$ of an ese is the identity we essentially have a prime event structure. This view is reinforced in our choice of maps. A map from ese $(P, \leq_P, \mathrm{Con}_P \equiv_P)$ to $(Q, \leq_Q, \mathrm{Con}_Q, \equiv_Q)$ is a partial function $f : P \rightharpoonup Q$ which *preserves* $\equiv$, *i.e.* if $p_1 \equiv_P p_2$ then either both $f(p_1)$ and $f(p_2)$ are undefined or both defined with $f(p_1) \equiv_Q f(p_2)$), such that for all $x \in \mathcal{C}(P)$ we have (i) the direct image $fx \in \mathcal{C}(Q)$, and (ii) $\forall p_1, p_2 \in x,\ f(p_1) \equiv_Q f(p_2) \implies p_1 \equiv_P p_2$. Maps compose as partial functions with the usual identities. Such maps preserve the concurrency relation. They are only assured to reflect causal dependency locally w.r.t. unambiguous configurations.

We regard two maps $f_1, f_2 : P \to Q$ as equivalent, and write $f_1 \equiv f_2$, iff they are equi-defined and yield equivalent results, *i.e.*if $f_1(p)$ is defined then so is $f_2(p)$ and $f_1(p) \equiv_Q f_2(p)$, and symmetrically. Composition respects $\equiv$: if $f_1, f_2 : P \to Q$ with $f_1 \equiv f_2$ and $g_1, g_2 : Q \to R$ with $g_1 \equiv g_2$, then $g_1 f_1 \equiv g_2 f_2$. Write $\mathcal{E}_{\equiv}$ for the category of ese's; it is *enriched* in the category of sets with equivalence relations – see [3] and Appendix A.

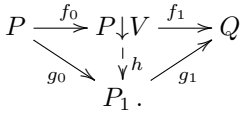
Ese's support a hiding operation. Let $(P, \leq, \mathrm{Con}_P, \equiv)$ be an ese. Let $V \subseteq P$ be a $\equiv$-closed subset of 'visible' events. Define the *projection* of P on V, to be $P{\downarrow}V =_{\mathrm{def}} (V, \leq_V, \mathrm{Con}_V, \equiv_V)$, where $v \leq_V v'$ iff $v \leq v'$ & $v, v' \in V$ and $X \in \mathrm{Con}_V$ iff $X \in \mathrm{Con}$ & $X \subseteq V$ and $v \equiv_V v'$ iff $v \equiv v'$ & $v, v' \in V$.

Hiding is associated with a factorisation of partial maps. Let f be a partial map from $(P, \leq_P, \mathrm{Con}_P, \equiv_P)$ to $(Q, \leq_Q, \mathrm{Con}_Q, \equiv_Q)$. Letting $V =_{\mathrm{def}} \{e \in E \mid f(e) \text{ is defined}\}$, the map f factors into the composition

$$P \xrightarrow{f_0} P{\downarrow}V \xrightarrow{f_1} Q$$

of f_0, a partial map of ese's taking $p \in P$ to itself if $p \in V$ and undefined otherwise, and f_1, a total map of ese's acting like f on V. We call f_1 the *defined part* of the partial map f. Because $\equiv$-equivalent maps share the same domain of definition, $\equiv$-equivalent maps will determine the same projection and $\equiv$-equivalent defined parts. The factorisation is characterised to within isomorphism by the following universal characterisation: for any factorisation $P \xrightarrow{g_0} P_1 \xrightarrow{g_1} Q$ where g_0 is partial and g_1 is total there is a (necessarily

total) unique map $h : P{\downarrow}V \to P_1$ such that we obtain the commuting diagram



The category $\mathcal{E}_{\equiv}$ of ese's supports hiding in the sense above.

5 Unfolding general event structures to ese's

We next show how replete general event structures embed in ese's as part of a (pseudo) reflection. This fixes the sense in which ese's extend the established model of general event structures in their treatment of parallel causes, while in addition supporting hiding. The relevant (pseudo) adjoint from $\mathcal{G}$ to $\mathcal{E}_{\equiv}$ is quite subtle and is a form of unfolding of a general event structure into an ese of its prime causes.

The pseudo functor arises as a right adjoint to a more obvious functor from $\mathcal{E}_{\equiv}$ to $\mathcal{G}$. Given an ese $(P, \leq, \mathrm{Con}, \equiv)$ we can construct a (replete) general event structure $ges(P) =_{\mathrm{def}} (E, \mathrm{Con}_E, \vdash)$ by taking

$$
\begin{array}{l}
E = P_{\equiv}, \text{ the equivalence classes under } \equiv, \\
X \in \mathrm{Con}_E \text{ iff } \exists Y \in \mathrm{Con},\ X = Y_{\equiv}\,, \text{ and} \\
X \vdash e \text{ iff } X \in \mathrm{Con} \ \&\ e \in E \ \&\ \exists p \in P,\ e = \{p\}_{\equiv} \ \&\ [p]_{\equiv} \subseteq X \cup \{e\}\,.
\end{array}
$$

The construction extends to a functor $ges : \mathcal{E}_{\equiv} \to \mathcal{G}$ as maps between ese's preserve $\equiv$; the functor takes a map $f : P \to Q$ of ese's to the map $ges(f) : ges(P) \to ges(Q)$ obtained as the partial function induced on equivalence classes. Less obvious is that there is a (pseudo) right adjoint to *ges*. Its construction relies on *extremal causal realisations* which provide us with an appropriate notion of minimal complete enabling of events in a general event structure; these furnish us with the prime causes from which to build the ese unfolding.

5.1 Causal realisations

Let $\mathcal{A}$ be a family of configurations with underlying set A. A *(causal) realisation* of $\mathcal{A}$ comprises a partial order $(E, \leq)$, its *carrier*, such that the set $\{e' \in E \mid e' \leq e\}$ is finite for all events $e \in E$, together with a function $\rho : E \to A$ for which the image $\rho x \in \mathcal{A}$ when x is a down-closed subset of E.

A map between realisations $(E, \leq), \rho$ and $(E', \leq'), \rho'$ is a partial surjective function $f : E \rightharpoonup E'$ which preserves down-closed subsets and satisfies $\rho(e) = \rho'(f(e))$ when $f(e)$ is defined. It is convenient to write such a map as $\rho \succeq^f \rho'$. Occasionally we shall write $\rho \succeq \rho'$, or the converse $\rho' \preceq \rho$, to mean there is a map of realisations from ρ to ρ'. Such a map factors into a "projection" followed by a total map

$$\rho \succeq_1^{f_1} \rho_0 \succeq_2^{f_2} \rho'\,,$$

where ρ_0 stands for the realisation $(E_0, \leq_0), \rho_0$ where $E_0 = \{r \in R \mid f(r) \text{ is defined}\}$ is the domain of definition of f, $\leq_0$ is the restriction of $\leq$, f_1 is the inverse relation to the inclusion $E_0 \subseteq E$, and f_2 is the total function $f_2 : E_0 \to E'$. We are using $\succeq_1$ and $\succeq_2$ to signify the two kinds of maps. Notice that $\succeq_1$-maps are reverse inclusions. Notice too that $\succeq_2$-maps are exactly the total maps of realisations. Total maps $\rho \succeq_2^f \rho'$ are precisely those functions f from the carrier of ρ to the carrier of ρ' which preserve down-closed subsets and satisfy $\rho = \rho' f$.

We shall say a realisation ρ is *extremal* when $\rho \succeq_2^f \rho'$ implies f is an isomorphism, for any realisation ρ'; it is called *prime extremal* when it in addition has a top element, *i.e.* its carrier contains an element which dominates all other elements in the carrier.

In the special case where $\mathcal{A}$ is the family of configurations of a prime event structure, it is easy to show that an extremal realisation ρ forms a bijection with a configuration of the event structure and that the order on the carrier coincides with causal dependency there; the prime extremals correspond to configurations of the form $[e]$ for some event e.

The construction is more interesting when $\mathcal{A}$ is the family of configurations of a general event structure A. In general, there is at most one map between extremal realisations. Hence extremal realisations of $\mathcal{A}$ under $\preceq$ form a preorder. The *order of extremal realisations* has as elements isomorphism classes of extremal realisations ordered according to the existence of a map between representatives of isomorphism classes. In fact:

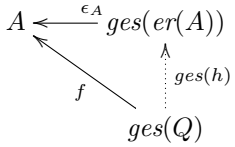
▶ **Theorem 1.** *The order of extremal realisations of $\mathcal{A}$ forms a prime-algebraic domain [5] with complete primes represented by the prime extremal realisations.*

The import of this theorem is that the order of extremal realisations is isomorphic to the configurations of a prime event structure ordered by inclusion. The event structure has events the prime extremal realisations; causal dependency the restriction of the order on extremal realisations; with consistency induced by compatibility there.

5.2 A pseudo adjunction

From Theorem 1, a general event structure A determines a prime event structure with events the prime extremal realisations of $\mathcal{C}^\infty(A)$ [5]. The top element of each prime extremal images to an event of A, providing a map from prime extremals to A. To get the ese-unfolding $er(A)$ of A we further endow the prime event structure with an equivalence, taking two prime extremals as equivalent if their top elements have the same image. Because equivalent prime extremals are sent to the same event of A, we determine a map $\epsilon_A : ges(er(A)) \to A$ of general event structures. It is the component of the counit of the adjunction at A. (See Appendix B for the proof and the detailed construction of er.)

▶ **Theorem 2.** *Let $A \in \mathcal{G}$. For all $f : ges(Q) \to A$ in $\mathcal{G}$, there is a map $h : Q \to er(A)$ in $\mathcal{E}_\equiv$ such that $f = \epsilon_A \circ ges(h)$ i.e. so the diagram below commutes:*

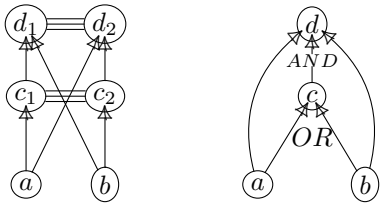


Moreover, if $h' : Q \to er(A)$ is a map in $\mathcal{E}_\equiv$ such that $f = \epsilon_A \circ ges(h')$, then $h' \equiv h$.

The theorem does not quite exhibit a standard adjunction, because the usual cofreeness condition specifying an adjunction is weakened to only having uniqueness up to $\equiv$. However the condition it describes does specify a simple case of *pseudo adjunction* between 2-categories – a set together with an equivalence relation is a very simple example of a category (see Appendix A). As a consequence, whereas with the usual cofreeness condition allows us to extend the right adjoint to arrows, so obtaining a functor, in this case following that same line will only yield a pseudo functor er as right adjoint: thus extended, er will only necessarily preserve composition and identities up to $\equiv$.

The pseudo adjunction from $\mathcal{E}_{\equiv}$ to $\mathcal{G}$ cuts down to a pseudo reflection (*i.e.* the counit ϵ is a natural isomorphism) when we restrict to the subcategory of $\mathcal{G}$ where all general event structures are replete. Its right adjoint provides a pseudo functor embedding replete general event structures (and so families of configurations) in ese's.

▶ **Example 3.** On the right we show a general event structure and on its left the ese which it unfolds to under *er*:



Although they don't appear in this example, it is possible to have extremal realisations in which an event depends on an event of the family having been enabled in two distinct ways – see Appendix B, Example 10. (Such phenomena will be disallowed in edc's.)

6 EDC'S

Our major motivation in developing and exploring ese's was in order to extend strategies with parallel causes while maintaining the central operation of hiding. What about the other operation key to the composition of strategies, *viz.* pullback?

It is well-known to be hard to construct limits such as pullback within prime event structures, so that we often rely on first carrying out the constructions in stable families, into which there is a coreflection from prime event structures. We might expect an analogous way to construct pullbacks or pseudo pullbacks in $\mathcal{E}_{\equiv}$.

6.1 Equivalence families

In fact, the pseudo adjunction from $\mathcal{E}_{\equiv}$ to $\mathcal{G}$ factors through a more basic pseudo adjunction to families of configurations which also bear an equivalence relation on their underlying sets. An *equivalence-family* (ef) is a family of configurations $\mathcal{A}$ with an equivalence relation $\equiv_A$ on its underlying set $\bigcup \mathcal{A}$. We can identify a family of configurations $\mathcal{A}$ with the ef $(\mathcal{A}, =)$, taking the equivalence to be simply equality on the underlying set. A map $f : (\mathcal{A}, \equiv_A) \to (\mathcal{B}, \equiv_B)$ between ef's is a partial function $f : A \rightharpoonup B$ between their underlying sets which preserves $\equiv$ so that

$$x \in \mathcal{A} \;\Rightarrow\; fx \in \mathcal{B} \;\&\; \forall a_1, a_2 \in x,\;\; f(a_1) \equiv_B f(a_2) \Rightarrow a_1 \equiv_A a_2\,.$$

Composition is composition of partial functions. We regard two maps $f_1, f_2 : (\mathcal{A}, \equiv_A) \to (\mathcal{B}, \equiv_B)$ as equivalent, and write $f_1 \equiv f_2$, iff they are equidefined and yield equivalent results. Composition respects $\equiv$. This yields a category of equivalence families $\mathcal{F}am_{\equiv}$ enriched in the category of sets with equivalence relations.

Clearly we can regard an ese $(P, \leq_P, \mathrm{Con}_P, \equiv_P)$ as an ef $(\mathcal{C}^\infty(P), \equiv_P)$ and a function which is a map of ese's as a map between the associated ef's, and this operation forms a functor. The functor has a pseudo right adjoint built from causal realisations in a very similar manner to *er*. The configurations of a general event structure form an ef with the identity relation as its equivalence. This operation is functorial and has a left adjoint which

collapses an ef to a general event structure in a similar way to *ges*; the adjunction is enriched in equivalence relations. In summary, the pseudo adjunction

$$\mathcal{E}_{\equiv} \underset{ges}{\overset{er}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} \mathcal{G}$$

factors into a pseudo adjunction followed by an adjunction

$$\mathcal{E}_{\equiv} \underset{\longrightarrow}{\overset{\longleftarrow}{\top}} \mathcal{F}am_{\equiv} \underset{\longrightarrow}{\overset{\longleftarrow}{\top}} \mathcal{G}\,.$$

$\mathcal{F}am_{\equiv}$ has pullbacks and pseudo pullbacks which are easy to construct. For example, let $f : \mathcal{A} \to \mathcal{C}$ and $g : \mathcal{B} \to \mathcal{C}$ be total maps of ef's. Assume $\mathcal{A}$ and $\mathcal{B}$ have underlying sets A and B. Define $D =_{\text{def}} \{(a, b) \in A \times B \mid f(a) \equiv_C g(b)\}$ with projections π_1 and π_2 to the left and right components. On D, take $d \equiv_D d'$ iff $\pi_1(d) \equiv_A \pi_1(d')$ and $\pi_2(d) \equiv_B \pi_2(d')$. Define a family of configurations of the *pseudo pullback* to consist of $x \in \mathcal{D}$ iff $x \subseteq D$ such that $\pi_1 x \in \mathcal{A}$ & $\pi_2 x \in \mathcal{B}$, and

$$\begin{aligned} &\forall d \in x \exists d_1, \cdots, d_n \in x,\ d_n = d\ \& \\ \forall i \leq n,\ &\pi_1\{d_1, \cdots, d_i\} \in \mathcal{A}\ \&\ \pi_2\{d_1, \cdots, d_i\} \in \mathcal{B}\,. \end{aligned}$$

The ef $\mathcal{D}$ with maps π_1 and π_2 is the pseudo pullback of f and g. It would coincide with pullback if $\equiv_C$ were the identity.

But unfortunately (pseudo) pullbacks in $\mathcal{F}am_{\equiv}$ don't provide us with (pseudo) pullbacks in $\mathcal{E}_{\equiv}$ because the right adjoint is only a pseudo functor: in general it will only carry pseudo pullbacks to bipullbacks. While $\mathcal{E}_{\equiv}$ does have bipullbacks (in which commutations and uniqueness are only up to the equivalence $\equiv$ on maps) it doesn't always have pseudo pullbacks or pullbacks – Appendix C. Whereas pseudo pullbacks and pullbacks are characterised up to isomorphism, bipullbacks are only characterised up to a weaker equivalence – that induced on objects by the equivalence on maps.[4] While we could develop strategies with parallel causes in the broad context of ese's, defining the composition of strategies via bipullbacks and hiding, doing so would mean that the composition of strategies that ensued was defined only up to equivalence and not isomorphism. Our definition of strategy-composition would be accordingly weaker in that its characterisation could only be up to equivalence.

6.2 Edc's defined

Fortunately there is a subcategory of $\mathcal{E}_{\equiv}$ which supports pullbacks and pseudo pullbacks, as well as hiding. Define $\mathcal{EDC}$ to be the subcategory of $\mathcal{E}_{\equiv}$ with objects ese's satisfying

$$p_1, p_2 \leq p\ \&\ p_1 \equiv p_2 \implies p_1 = p_2\,.$$

We call such objects *event structures with disjunctive causes* (edc's). In an edc an event can't causally depend on two distinct prime causes of a common disjunctive event, and so rules out realisations such as that mentioned in Example 10. In general, within $\mathcal{E}_{\equiv}$ we lose the local injectivity property that we're used to seeing for maps of event structures; the maps of event structures are injective from configurations, when defined. However for $\mathcal{EDC}$ we recover local injectivity w.r.t. prime configurations, of form $[p]$: if $f : P \to Q$ is a map in $\mathcal{EDC}$, then

$$p_1, p_2 \leq_P p\ \&\ f(p_1) = f(p_2) \implies p_1 = p_2\,.$$

[4] Objects P and Q are equivalent iff there are two maps $f : P \to Q$, $g : Q \to P$ s.t. $gf \equiv \mathrm{id}_P$ and $fg \equiv \mathrm{id}_Q$.

The factorisation property associated with hiding in $\mathcal{E}_{\equiv}$ is inherited by $\mathcal{EDC}$.

As regards (pseudo) pullbacks, we are fortunate in that the complicated pseudo adjunction between ese's and ef's restricts to a much simpler (pseudo) adjunction, in fact a coreflection, between edc's and *stable* ef's. In an equivalence family $(\mathcal{A}, \equiv_A)$ say a configuration $x \in \mathcal{A}$ is *unambiguous* iff $\forall a_1, a_2 \in x,\ a_1 \equiv_A a_2 \implies a_1 = a_2$. An equivalence family $(\mathcal{A}, \equiv_A)$, with underlying set of events A, is *stable* iff it satisfies

$$\forall x, y, z \in \mathcal{A},\ x, y \subseteq z \ \&\ z \text{ is unambiguous } \Rightarrow\ x \cap y \in \mathcal{A}\,, \text{ and}$$
$$\forall a \in A, x \in \mathcal{A},\ a \in x \Rightarrow \exists z \in \mathcal{A},\ z \text{ is unambiguous } \&\ a \in z \subseteq x\,.$$

In effect a stable equivalence family contains a stable subfamily of unambiguous configurations out of which all other configurations are obtainable as unions. Local to any unambiguous configuration x there is a partial order on its events $\leq_x$: each $a \in x$ determines a *prime configuration*

$$[a]_x =_{\text{def}} \bigcap \{y \in \mathcal{A} \mid a \in y \subseteq x\}\,,$$

the minimum set of events on which a depends within x; taking $a \leq_x b$ iff $[a]_x \subseteq [b]_x$ defines causal dependency between $a, b \in x$. Write $\mathcal{SFam}_{\equiv}$ for the subcategory of stable ef's.

(Pseudo) pullbacks in stable ef's are obtained from those in ef's simply by restricting to those configurations which are unions of unambiguous configurations. The configurations of an edc with its equivalence are easily seen to form a stable ef providing a full and faithful embedding of $\mathcal{EDC}$ in $\mathcal{SFam}_{\equiv}$. The embedding has a right adjoint Pr. It is built out of prime extremals but we can take advantage of the fact that in a stable ef unambiguous prime extremals have the simple form of prime configurations. From a stable ef $(\mathcal{A}, \equiv_A)$ we produce an edc $\Pr(\mathcal{A}, \equiv_A) =_{\text{def}} (P, \text{Con}, \leq, \equiv)$ in which P comprises the prime configurations with

$$[a]_x \equiv [a']_{x'} \text{ iff } a \equiv_A a'\,,$$
$$Z \in \text{Con iff } Z \subseteq P \ \&\ \bigcup Z \in \mathcal{F}\,, \text{ and}$$
$$p \leq p' \text{ iff } p, p' \in P \ \&\ p \subseteq p'\,.$$

The adjunction is enriched in the sense that its natural bijection preserves and reflects the equivalence on maps:

$$\mathcal{EDC} \underset{\top}{\overset{\Pr}{\leftarrow\!\rightarrow}} \mathcal{SFam}_{\equiv}$$

We can now obtain a (pseudo) pullback in edc's by first forming the (pseudo) pullback of the stable ef's obtained as their configurations and then taking its image under the right adjoint Pr. We now have the constructions we need to support strategies based on edc's.

6.3 Coreflective subcategories of edc's

$\mathcal{EDC}$ is a coreflective subcategory of $\mathcal{E}_{\equiv}$; the right adjoint simply cuts down to those events satisfying the edc property. In turn $\mathcal{EDC}$ has a coreflective subcategory $\mathcal{E}^0_{\equiv}$ comprising those edc's which satisfy

$$\{p_1, p_2\} \in \text{Con} \ \&\ p_1 \equiv p_2 \implies p_1 = p_2\,.$$

Consequently its maps are traditional maps of event structures which preserve the equivalence. We derive adjunctions

$$\mathcal{E}^0_{\equiv} \underset{\top}{\leftarrow\!\rightarrow} \mathcal{EDC} \underset{\top}{\leftarrow\!\rightarrow} \mathcal{E}_{\equiv} \underset{ges}{\overset{er}{\leftarrow\!\rightarrow}} \mathcal{G}\,.$$

Note the last is only a pseudo adjunction. Consequently we obtain a pseudo adjunction from $\mathcal{E}^0_{\equiv}$, the category of prime event structures with equivalence relations and general event structures – this makes good the promise of Section 2.3. Inspecting the composite of the last two adjunctions, we also obtain the sense in which replete general event structures embed via a reflection in edc's.

There is an obvious 'inclusion' functor from the category of prime event structures $\mathcal{E}$ to the category $\mathcal{EDC}$: it extends an event structure with the identity equivalence. Regarding $\mathcal{EDC}$ as a plain category, so dropping the enrichment by equivalence relations, the 'inclusion' functor $\mathcal{E} \hookrightarrow \mathcal{EDC}$ has a right adjoint, *viz.* the forgetful functor which given an edc $P = (P, \leq, \mathrm{Con}, \equiv)$ produces an event structure $P_0 = (P, \leq, \mathrm{Con}')$ by dropping the equivalence $\equiv$ and modifying the consistency relation to:

$$X \in \mathrm{Con}' \text{ iff } X \subseteq P \;\&\; X \in \mathrm{Con} \;\&\; p_1 \not\equiv p_2, \text{ for all } p_1, p_2 \in X\,.$$

The configurations of P_0 are the unambiguous configurations of P. The adjunction is a coreflection because the inclusion functor is full. Of course it is not the case that the adjunction is enriched: the natural bijection of the adjunction cannot respect the equivalence on maps; it cannot compose with the pseudo adjunction from $\mathcal{EDC}$ to $\mathcal{G}$ to yield a pseudo adjunction from $\mathcal{E}$ to $\mathcal{G}$.

Despite this the adjunction from $\mathcal{E}$ to $\mathcal{EDC}$ has many useful properties. Of importance for us is that the functor forgetting equivalence will preserve all limits and especially pullbacks. It is helpful in relating composition of edc-strategies to the composition of strategies based on prime event structures in [8]. In composing strategies in edc's we shall only be involved with pullbacks of maps $f : A \to C$ and $g : B \to C$ of edc's. (When C is essentially an event structure, *i.e.* an edc in which the equivalence is the identity relation, the construction of such pullbacks coincides with that of pseudo pullbacks.) While this does not entail that composition of strategies is preserved by the forgetful functor – because the forgetful functor does not commute with hiding – it will give us a strong relationship, expressed as a map, between composition of the two kinds of strategies (based on edc's and based on prime event structures) after and before applying the forgetful functor. This has been extremely useful in key proofs of the next section, in importing results about concurrent strategies from [8].

7 Probabilistic strategies based on edc's

The ground is prepared for a general definition of distributed probabilistic strategies, based on edc's. The development follows the same lines as that of probabilistic concurrent strategies [8, 11], to which we refer the reader, and can only be sketched briefly here.

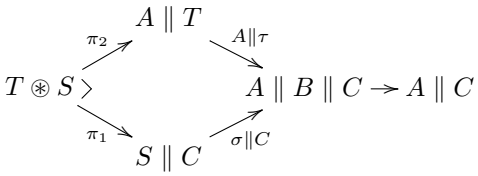
An *edc with polarity* comprises $(P, \leq_P, \mathrm{Con}_P, \equiv, pol)$, an edc $(P, \leq_P, \mathrm{Con}_P, \equiv)$ in which each element $p \in P$ carries a polarity $pol(p)$ which is $+$ or $-$, according as it represents a move of Player or Opponent, and where the equivalence relation $\equiv$ respects polarity. A *map* of edc's with polarity is a map of the underlying edc's which preserves polarity when defined. The adjunctions of the previous section are undisturbed by adding polarity.

A *game* is represented by an edc with polarity. There are two fundamentally important operations on two-party games. One is that of forming the *dual* game. On a game A this amounts to reversing the polarities of events to produce the dual $A^\perp$. The other operation, a *simple parallel composition* $A\|B$, is achieved on games A and B by simply juxtaposing them, ensuring a finite subset of events is consistent if its overlaps with the two games are individually consistent.

A *pre-strategy* in a game A is a total map $\sigma : S \to A$ of edc's with polarity. A pre-strategy from a game A to a game B is a pre-strategy in the game $A^\perp\|B$. A map $f : \sigma \Rightarrow \sigma'$ of

pre-strategies $\sigma : S \to A$ and $\sigma' : S' \to A$ is a map $f : S \to S'$ s.t. $\sigma = \sigma' f$; this determines isomorphism of pre-strategies. The map is *rigid* if it preserves causal dependency.

Two edc pre-strategies $\sigma : S \to A^{\perp} \| B$ and $\tau : T \to B^{\perp} \| C$ compose via *pullback* and *hiding* – with parallel causes, the key features driving our search for edc's. Ignoring polarities, the composite partial map



has defined part, yielding the composition $\tau \odot \sigma : T \odot S \to A^{\perp} \| C$ once we reinstate polarities. (The partial map from $A \| B \| C$ to $A \| C$ acts as the identity but for being undefined on B.)

The copycat strategy comprises $\text{𝒸}_A : \text{𝒞}_A \to A^{\perp} \| A$ where 𝒞_A is obtained by adding extra causal dependencies to $A^{\perp} \| A$ so that any Player move in either component causally depends on its copy, an Opponent move, in the other [8].

In general, copycat may not be an identity w.r.t. composition. However, copycat acts as identity precisely on an edc pre-strategy $\sigma : S \to A$ which is an *edc strategy*, capturing the sense in which Player cannot influence Opponent beyond the constraints of the game:

(i) the image $\sigma_0 : S_0 \to A_0$ of σ (under the right adjoint to the inclusion of event structures in edc's) is a strategy of concurrent games, *i.e.* is *receptive* and *innocent*, as in [8];[5] and

(ii) $s_1 \equiv_S s_2 \iff \sigma(s_1) \equiv_A \sigma(s_2)$, for all $s_1, s_2 \in S$; with

(iii) $x \overset{s}{\text{—}\!\subset} z \;\&\; x \overset{s'}{\text{—}\!\subset} z' \;\&\; pol(s) = - \;\&\; \sigma z \uparrow \sigma z' \implies z \uparrow z'$.

▶ **Theorem 4.** *When $\sigma : S \to A$ is an edc pre-strategy, $\sigma \cong \text{𝒸}_A \odot \sigma$ iff σ is an edc strategy.*

A *probabilistic edc strategy* in a game A, is an edc strategy $\sigma : S \to A$ together with a configuration-valuation which endows S with probability, while taking account of the fact that in the strategy Player can't be aware of the probabilities assigned by Opponent. We should restrict to race-free games, precisely those for which copycat is deterministic, so that we have probabilistic identity strategies; it follows that S is race-free. A configuration-valuation extends the definition of probabilistic event structure [11] with an axiom (lmc) which implies the Limited Markov Condition, LMC, of the introduction. Precisely, a *configuration-valuation* is a function $v : \mathcal{C}(S) \to [0, 1]$ which is: *(normalized)* $v(\emptyset) = 1$; and satisfies:

- *(lmc)* $v(x) = v(y)$ when $x \subseteq^{-} y$ for finite configurations x, y of S
- *(+ve drop condition)* $d_v[y; x_1, \cdots, x_n] \geq 0$ when $y \subseteq^{+} x_1, \cdots, x_n$ for finite configurations.

The 'drop' function, $d_v[y; x_1, \cdots, x_n] =_{\text{def}} v(y) - \sum_I (-1)^{|I|+1} v(\bigcup_{i \in I} x_i)$, where the index I ranges over nonempty $I \subseteq \{1, \cdots, n\}$ such that the union $\bigcup_{i \in I} x_i \in \mathcal{C}(S)$. Above we use $x \subseteq^{-} y$, and $x \subseteq^{+} y$, to mean inclusion in which all the intervening events have the indicated polarity.

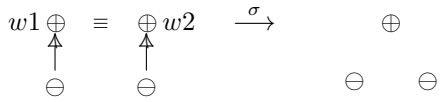
When there are no Opponent moves in S, a configuration-valuation corresponds to a continuous valuation on the Scott-open sets of $\mathcal{C}^{\infty}(S)$ and determines a probability distribution on the Borel sets; then $v(x)$ is $\text{Prob}(x)$, the probability that the result includes the events of the finite configuration x [11]. When S has Opponent moves, the reading of a

[5] A total map of event structures with polarity $\sigma : S \to A$ is *receptive* if $\sigma x \overset{a}{\text{—}\!\subset} \;\&\; pol(a) = -$ implies $\exists! s,\; x \overset{s}{\text{—}\!\subset} \;\&\; \sigma(s) = a$. It is *innocent* if $s \rightarrowtriangle s'$ with $pol(s) = +$ or $pol(s') = -$ implies $\sigma(s) \rightarrowtriangle \sigma(s')$.

configuration-valuation involves conditional probabilities. When $x \subseteq^+ y$ in $\mathcal{C}(S)$, provided $v(x) \neq 0$, the conditional probability of Player making moves $y \setminus x$ given x, is expressed by $\mathrm{Prob}(y \mid x) = v(y)/v(x)$. Because S is race-free, this reading, with (lmc), ensures we obtain LMC directly.

The composition above extends to probabilistic edc strategies. Assuming σ and τ have configuration-valuations v_S and v_T their composition $\tau \odot \sigma$ has configuration-valuation $v(x) =_{\mathrm{def}} v_S([x]_S).v_T([x]_T)$ for x a finite configuration of $T \odot S$; the configuration $[x]_S$ is the S-component in $\mathcal{C}(S)$ of the projection $\pi_1[x]$, and $[x]_T$ the T-component of $\pi_2[x]$. The proof that v is a configuration-valuation relies heavily on properties of "drop" functions.

We obtain a bicategory of probabilistic edc strategies which support parallel causes; its 2-cells are rigid maps of strategies which relate configuration-valuations across 2-cells via a 'push-forward' result – see Appendix D. It has a sub-bicategory of deterministic edc strategies analogous to that of [8]. But now there are deterministic strategies with parallel causes, including the strategy sketched informally in the introduction in which Player makes a move iff Opponent makes one or more of their moves:



Similarly, there are now deterministic strategies for "parallel or" and McCarthy's amb [4].

▶ **Example 5.** Recall the game of the introduction. In the edc strategy drawn above, individual success of the two watchers $w1$ and $w2$ may be associated with probabilities $p_1 \in [0,1]$ and $p_2 \in [0,1]$, respectively, and their joint success with $q \in [0,1]$ provided they form a configuration-valuation v. In other words, $v(x) = p_1$ if x contains $w1$ and not $w2$; $v(x) = p_2$ if x contains $w2$ and not $w1$; and $v(x) = q$ if x contains both $w1$ and $w2$; $v(x) = 1$ otherwise; and $p_1 + p_2 - q \leq 1$, in order to satisfy the $+$-drop condition. To enliven this a little we might imagine the two watchers have a drinking problem and the correlation depends on whether they are sharing from a common bottle: if they had their own bottles we might imagine the drunken unreliability of one independent of that of the other, so $q = p_1.p_2$; as good friends sharing from a common bottle their drunkeness might correlate, so $p_1 = p_2 = q$.

Acknowledgments. Thanks to the referees, to Simon Castellan, Pierre Clairambault, Mai Gehrke, Jonathan Hayman and Martin Hyland for advice and encouragement, to ENS Paris for supporting Marc de Visme's internship and to the ERC for Advanced Grant ECSYM.

References

1 Ioana Cristescu. *Operational and denotational semantics for the reversible pi-calculus.* PhD thesis, PPS, Université Paris Diderot, 2015.
2 G. M. Kelly. *Basic concepts of enriched category theory.* LNM 64. CUP, 1982.
3 Y. Kinoshita and J. Power. Category theoretic structure of setoids. *Theor. Comput. Sci.*, 546, 2014.
4 John McCarthy. A basis for a mathematical theory of computation. In P. Brafford and D. Hirschberg, editors, *Computer Programming and Formal Systems.* North-Holland, 1963.
5 Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains. *TCS*, 13:85–108, 1981.
6 Judea Pearl. *Causality.* CUP, 2013.
7 John Power. 2-categories. *BRICS Notes Series NS-98-7*, 1998.
8 Silvain Rideau and Glynn Winskel. Concurrent strategies. In *LICS*, 2011.

9 Glynn Winskel. *Events in computation*. PhD thesis, University of Edinburgh, 1980.
10 Glynn Winskel. Event structures. In *Advances in Petri Nets*, LNCS 255, 1986.
11 Glynn Winskel. Distributed probabilistic and quantum strategies. *ENTCS 298*, 2013.
12 Glynn Winskel. Event structures, stable families and concurrent games, 2016. URL: `https://www.cl.cam.ac.uk/~gw104/ecsym-notes.pdf`.
13 Glynn Winskel and Mogens Nielsen. *Handbook of Logic in Computer Science 4*, chapter Models for Concurrency, pages 1–148. OUP, 1995.

A Equiv-enriched categories

Here we explain in more detail what we mean when we say "enriched in the category of of sets with equivalence relations" and employ terms such as "enriched adjunction," "pseudo adjunction" and "pseudo pullback." The classic text on enriched categories is [2], but for this paper the articles [3] and [7] provide short, accessible introductions to the notions we use from Equiv-enriched categories and 2-categories, respectively.

Equiv is the category of *equivalence relations*. Its objects are $(A, \equiv_A)$ comprising a set A on which there is an equivalence relation $\equiv_A$. Its maps $f : (A, \equiv_A) \to (B, \equiv_B)$ are total functions $f : A \to B$ which preserve equivalence.

We shall use some basic notions from enriched category theory [2]. We shall be concerned with categories enriched in Equiv, called Equiv-enriched categories, in which the homsets possess the structure of equivalence relations, respected by composition [3]. This is the sense in which we say categories are enriched in (the category of) equivalence relations. We similarly borrow the concept of an Equiv-enriched functor between Equiv-enriched categories which preserve equivalence in acting on homsets. An Equiv-enriched adjunction is a usual adjunction in which the natural bijection preserves and reflects equivalence.

Because an object in Equiv can be regarded as a (very simple) category, we can regard Equiv-enriched categories as (very simple) 2-categories to which notions from 2-categories apply [7].

A *pseudo functor* between Equiv-enriched categories is like a functor but the usual laws only need hold up to equivalence. A *pseudo adjunction* (or biadjunction) between 2-categories permits a weakening of the usual natural isomorphism between homsets, now also categories, to a natural equivalence of categories. In the special case of a pseudo adjunction between Equiv-enriched categories the equivalence of homset categories amounts to a pair of $\equiv$-preserving functions whose compositions are $\equiv$-equivalent to the identity function. With traditional adjunctions by specifying the action of one adjoint solely on objects we determine it as a functor; with pseudo adjunctions we can only determine it as a pseudo functor – in general a pseudo adjunction relates two pseudo functors. Pseudo adjunctions compose in the expected way. An Equiv-enriched adjunction is a special case of a 2-adjunction between 2-categories and a very special case of pseudo adjunction. In this article there are many cases in which we compose an Equiv-enriched adjunction with a pseudo adjunction to obtain a new pseudo adjunction.

Similarly we can specialise the notions pseudo pullbacks and bipullbacks from 2-categories to Equiv-enriched categories. Let $f : A \to C$ and $g : B \to C$ be two maps in an Equiv-enriched category. A *pseudo pullback* of f and g is an object D and maps $p : D \to A$ and $q : D \to B$ such that $f \circ p \equiv g \circ q$ which satisfy the further property that for any D' and maps $p' : D' \to A$ and $q' : D' \to B$ such that $f \circ p' \equiv g \circ q'$, there is a unique map $h : D' \to D$ such that $p' = p \circ h$ and $q' = q \circ h$. There is an obvious weakening of pseudo pullbacks to the situation in which the uniqueness is replaced by uniqueness up to $\equiv$ and the equalities by $\equiv$ – these are simple special cases of bilimits called *bipullbacks*.

Right adjoints in a 2-adjunction preserve pseudo pullbacks whereas right adjoints in a pseudo adjunction are only assured to preserve bipullbacks.

B The proof of Theorem 2

Here we fill in some details of the proof of Theorem 2 providing a pseudo right adjoint to the functor $ges : \mathcal{E}_{\equiv} \to \mathcal{G}$: the functor ges quotients an ese down to a general event structure; its right adjoint er constructs an ese out of the prime extremal realisations of a general event structure. Note the adjunction is not an equivalence: whereas it does cut down to a reflection from $\mathcal{G}$ to $\mathcal{E}_{\equiv}$, where the counit is an isomorphism, the unit is not an isomorphism.

The right adjoint $er : \mathcal{G} \to \mathcal{E}_{\equiv}$ is defined on objects as follows. Let A be a general event structure. Define $er(A) = (P, \mathrm{Con}_P, \leq_P, \equiv_P)$ where

- P consists of a choice from within each isomorphism class of the prime extremals p of $\mathcal{C}^\infty(A)$ – we write $top_A(p)$ for the image of the top element in A;
- Causal dependency $\leq_P$ is $\preceq$ on P;
- $X \in \mathrm{Con}_P$ iff $X \subseteq_{\mathrm{fin}} P$ and $top_A[X] \in \mathcal{C}^\infty(A)$ – the set $[X]$ is the $\leq_P$-downwards closure of X;
- $p_1 \equiv_P p_2$ iff $p_1, p_2 \in P$ and $top_A(p_1) = top_A(p_2)$.

▶ **Proposition 6.** *The configurations of P, ordered by inclusion, are order-isomorphic to the order of extremal realisations of $\mathcal{C}^\infty(A)$: an extremal realisation ρ corresponds, up to isomorphism, to the configuration $\{p \in P \mid p \preceq \rho\}$ of P; conversely, a configuration x of P corresponds to an extremal realisation $top_A : x \to A$ with carrier $(x, \preceq)$, the restriction of the order of P to x.*

Proof. See the proof of Proposition 16.17 of the ECSYM Notes [12]. ◀

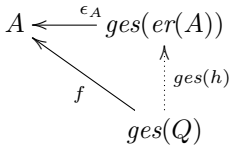
Theorem 1 of the main text, asserting the prime algebraicity of the order of extremal realisations, follows as an immediate corollary of the above proposition.

In defining the right adjoint we rely on the fact that any realisation of a family of configurations can be coarsened to an extremal realisation.

▶ **Lemma 7.** *For any realisation ρ there is an extremal realisation ρ' with $\rho \succeq^f_2 \rho'$.*

Proof. See the proof of Lemma 16.4 of the ECSYM Notes [12]. ◀

▶ **Theorem 2** (restated). *Let $A \in \mathcal{G}$. For all $f : ges(Q) \to A$ in $\mathcal{G}$, there is a map $h : Q \to er(A)$ in $\mathcal{E}_{\equiv}$ such that $f = \epsilon_A \circ ges(h)$ i.e. so the diagram below commutes:*



Moreover, if $h' : Q \to er(A)$ is a map in $\mathcal{E}_{\equiv}$ such that $f = \epsilon_A \circ ges(h')$, then $h' \equiv h$.

Proof. The component of the counit of the adjunction at A is given by the function ϵ_A taking $\{p\}_{\equiv}$ to $top_A(p)$; it determines a map $\epsilon_A : ges(er(A)) \to A$ of general event structures.

Let $Q = (Q, \mathrm{Con}_Q, \leq_Q, \equiv_Q)$ be an ese and $f : ges(Q) \to A$ a map in $\mathcal{G}$. We shall define a map $h : Q \to er(A)$ s.t. $f = \epsilon_A \circ ges(h)$. Notice that $\epsilon_A \circ ges(h)(\{q\}_{\equiv_Q}) = top_A(h(q))$, so the requirement that $f = \epsilon_A \circ ges(h)$ amounts to

$$f(\{q\}_{\equiv_Q}) = top_A(h(q)), \quad \text{for all } q \in Q.$$

We define the map $h : Q \to er(A)$ by induction on the depth of Q. The depth of an event in an event structure is the length of a longest $\leq$-chain up to it – so an initial event has depth 1. We take the depth of an event structure to be the maximum depth of its events. (Because of our reliance on Lemma 7, the proof of which uses the axiom of choice, we using the axiom of choice implicitly.)

Assume inductively that $h^{(n)}$ defines a map from $Q^{(n)}$ to $er(A)$ where $Q^{(n)}$ is the restriction of Q to depth below or equal to n such that $f^{(n)}$ the restriction of f to $Q^{(n)}$ satisfies $f^{(n)} = \epsilon_A \circ ges(h^{(n)})$. (In particular, $Q^{(0)}$ is the empty ese and $h^{(0)}$ the empty function.) Then, by Proposition 6, any configuration x of $Q^{(n)}$ determines an extremal realisation $\rho_x : h^{(n)}x \to A$ with carrier $(h^{(n)}x, \preceq)$.

Suppose $q \in Q$ has depth $n+1$. If $f(q)$ is undefined take $h^{(n+1)}(q)$ to be undefined. Otherwise, note there is an extremal realisation $\rho_{[q)}$ with carrier $(h[q), \preceq)$. Extend $\rho_{[q)}$ to a realisation $\rho^{\top}_{[q)}$ with carrier that of $\rho_{[q)}$ with a new top element $\top$ adjoined, and make $\rho^{\top}_{[q)}$ extend the function $\rho_{[q)}$ by taking $\top$ to $f(q)$. By Lemma 7, there is an extremal realisation ρ such that $\rho^{\top}_{[q)} \succeq_2 \rho$. Because $\rho_{[q)}$ is extremal $\rho_{[q)} \preceq_1 \rho$, so ρ only extends the order of $\rho_{[q)}$ with extra dependencies of $\top$. (For notational simplicity we identify the carrier of ρ with the set $h[q) \cup \{\top\}$.) Project ρ to the extremal with top $\top$. Define this to be the value of $h^{(n+1)}(q)$. In this way, we extend $h^{(n)}$ to a partial function $h^{(n+1)} : Q^{(n+1)} \to er(A)$ such that $f^{(n+1)} = \epsilon_A \circ ges(h^{(n+1)})$. In showing that $h^{(n+1)}$ is a map we rely on f being a map.

Defining $h = \bigcup_{n\in\omega} h^{(n)}$ we obtain a map $h : Q \to er(A)$ such that $f = \epsilon_A \circ ges(h)$.

Suppose $h' : Q \to er(A)$ is a map sush that $f = \epsilon_A \circ ges(h')$. Then, for any $q \in Q$,

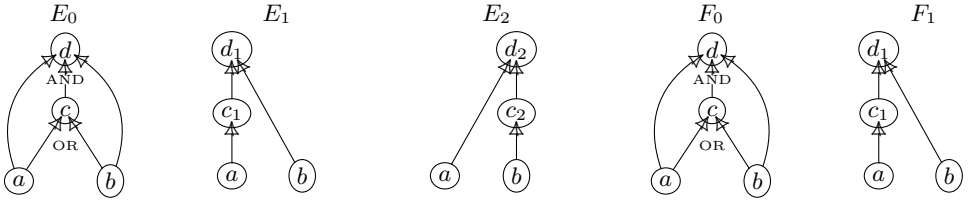
$$top_A(h'(q)) = \epsilon_A \circ ges(h')(\{q\}_{\equiv_Q}) = f(\{q\}_{\equiv_Q}) = \epsilon_A \circ ges(h)(\{q\}_{\equiv_Q}) = top_A(h(q))\,,$$

so $h'(q) \equiv h(q)$ in $er(A)$. Thus $h' \equiv h$. ◀

A configuration $x \in \mathcal{F}$, of a family of configurations $\mathcal{F}$, is *irreducible* iff there is a necessarily unique $e \in x$ such that $\forall y \in \mathcal{F},\ e \in y \subseteq x$ implies $y = x$. Irreducibles coincide with complete join irreducibles w.r.t. the order of inclusion. It is tempting to think of irreducibles as representing minimal complete enablings. But, as sets, irreducibles both (1) lack sufficient structure: in the formulation we are led to of minimal complete enabling as prime extremal realisations, several prime realisations can have the same irreducible as their underlying set; and (2) are not general enough: there are prime realisations whose underlying set is not an irreducible.

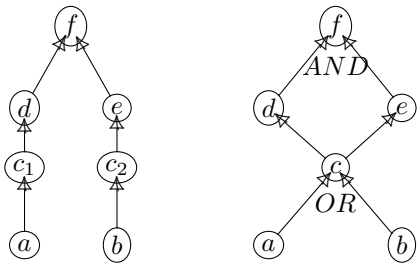
We provide examples illustrating the nature of extremal realisations. In the examples it is convenient to describe families of configurations by general event structures, taking advantage of the economic representation they provide.

▶ **Example 8.** This and the following example shows that prime extremal realisations do not correspond to irreducible configurations. Here, we show a general event structure E_0 with irreducible configuration $\{a, b, c, d\}$ and two prime extremals E_1 and E_2 with tops d_1 and d_2 which both have the same irreducible configuration $\{a, b, c, d\}$ as their image. The lettering indicates the functions associated with the realisations, *e.g.* events d_1 and d_2 in the partial orders map to d in the general event structure.



▶ **Example 9.** On the other hand there are prime extremal realisations of which the image is not an irreducible configuration. We consider the general event structure F_0. The prime extremal F_1 describes a situation where d is enabled by b and c is enabled by a. It has image the configuration $\{a, b, c, d\}$ which is not irreducible, being the union of the two configurations $\{a\}$ and $\{b, c, d\}$.

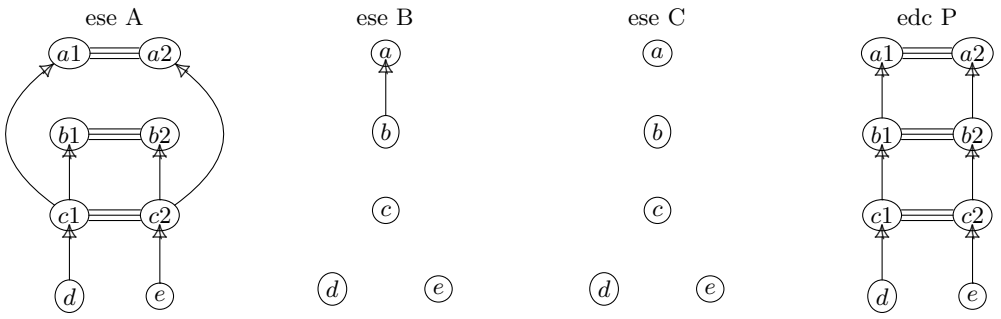
▶ **Example 10.** It is possible to have extremal realisations in which an event depends on an event of the family having been enabled in two distinct ways, as in the following prime extremal realisation, on the left.



The extremal describes the event f being enabled by d and e where they are in turn enabled by different ways of enabling c. Such phenomena are disallowed in edc's.

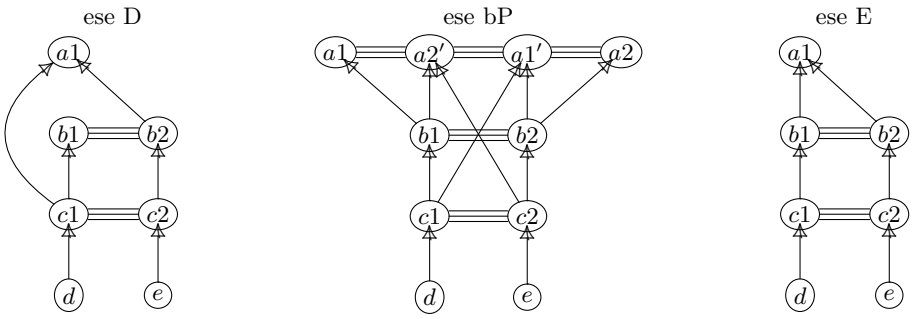
C On (pseudo) pullbacks of ese's

We show that the enriched category of ese's $\mathcal{E}_{\equiv}$ does not always have pullbacks and pseudo pullbacks of maps $f : A \to C$ and $g : B \to C$, the reason why we use the subcategory $\mathcal{EDC}$, which does, as a foundation on which to develop strategies with parallel causes. It suffices to exhibit the lack of pullbacks when C is an (ese of an) event structure as then pullbacks and pseudo pullbacks coincide. Take A, B, C as below, with the obvious maps $f : A \to C$ and $g : B \to C$ (given by the lettering). In fact, A and B are edc's.



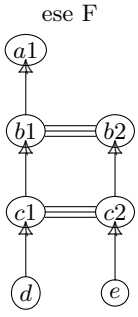
The pullback in edc's $\mathcal{EDC}$ does exist and is given by P with the obvious projection maps. However this is not a pullback in $\mathcal{E}_{\equiv}$. Consider the ese D with the obvious total maps to A and B; they form a commuting square with f and g. This cannot factor through P: event $b2$ has to be mapped to $b2$ in P, but then $a1$ cannot be mapped to $a1$ (it wouldn't yield a

map) nor to $a2$ (it would violate commutation required of a pullback).



There is a bipullback bP got by applying the pseudo functor er to the pullback in ef's. But this is not a pullback because in the ese E the required mediating map is not unique in that $a1$ can go to either $a1$ or $a1'$. In fact, there is no pullback of f and g. To show this we use the additional ese F.

Suppose Q with projection maps to A and B were a pullback of f and g in $\mathcal{E}_{\equiv}$. Consider the three ese's D, E and F with their obvious maps to A and B; in each case they form a commuting square with f and g. There are three unique maps $h_D : D \to Q$, $h_E : E \to Q$, and $h_F : F \to Q$ such that the corresponding pullback diagrams commute. We remark that there are also obvious maps $k_D : E \to D$ and $k_F : E \to F$ (given by the lettering) which commute with the maps to the components A and B. By uniqueness, we have $h_D \circ k_D = h_E = h_F \circ k_F$, so we have $h_D(a1) = h_F(a1)$. From the definition of the maps, the event $h_D(a1) = h_F(a1)$ has at most one $\leq$-predecessor in Q which is sent to b in C (as D only has one). Because of the projection to B, it has at least one (as B has one). So the event $h_D(a1) = h_F(a1)$ has exactly one predecessor which is sent to b. From the definition of maps, this event is $h_D(b2)$ which equals $h_F(b1)$. But $h_D(b2)$ cannot equal $h_F(b1)$ as they go to two different events of A – a contradiction.



Hence there can be no pullback of f and g in $\mathcal{E}_{\equiv}$. (By adding intermediary events, we would encounter essentially the same example in the composition, before hiding, of strategies if they were to be developed within the broader category of ese's.)

D The bicategory of probabilistic edc strategies

We obtain a bicategory of probabilistic edc strategies in which objects are race-free games. Maps are probabilistic edc strategies, composition that of strategies and identities are given by copycat strategies, which for race-free games are deterministic, so permit configuration-valuations which are constantly 1.

The 2-cells of the bicategory require consideration. Whereas we can always "push forward" a probability measure from the domain to the codomain of a measurable function this is not true generally for configuration-valuations involving Opponent moves. However:

▶ **Theorem 11.** *Let $f : \sigma \Rightarrow \sigma'$ be a rigid 2-cell between edc strategies $\sigma : S \to A$ and $\sigma' : S' \to A$. Let v be a configuration-valuation on S. Defining, for $y \in \mathcal{C}(S')$,*

$$(fv)(y) =_{\text{def}} \sup_X \sum_{\emptyset \neq Z \subseteq X \,\&\, Z\uparrow} (-1)^{|Z|+1} v(\bigcup Z)$$

as X ranges over finite subsets of $\{x \in \mathcal{C}(S) \mid y = fx\}$, yields a configuration-valuation fv of S' – the push-forward *of v.*

A 2-cell from σ, v to σ', v' is a rigid 2-cell $f : \sigma \Rightarrow \sigma'$ of edc strategies for which the push-forward fv is pointwise less than or equal to v', *i.e.*

$$(fv)(x') \leq v'(x')\,,$$

for all configurations $x' \in \mathcal{C}(S')$. Vertical composition of 2-cells is their usual composition. Horizontal composition is given by composition $\odot$, which extends to a functor on 2-cells via the universality of pullback and the factorisation property of hiding.

An Algebraic Approach to Valued Constraint Satisfaction*

Rostislav Horčík[1], Tommaso Moraschini[2], and Amanda Vidal[3]

1 **Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic**
horcik@cs.cas.cz
2 **Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic**
moraschini@cs.cas.cz
3 **Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic**
amanda@cs.cas.cz

Abstract

We study the complexity of the valued CSP (VCSP, for short) over arbitrary templates, taking the general framework of integral bounded linearly order monoids as valuation structures. The class of problems considered here subsumes and generalizes the most common one in VCSP literature, since both monoidal and lattice conjunction operations are allowed in the formulation of constraints. Restricting to locally finite monoids, we introduce a notion of polymorphism that captures the pp-definability in the style of Geiger's result. As a consequence, sufficient conditions for tractability of the classical CSP, related to the existence of certain polymorphisms, are shown to serve also for the valued case. Finally, we establish the dichotomy conjecture for the VCSP, modulo the dichotomy for classical CSP.

1998 ACM Subject Classification F.4.1 Mathematical Logic; G.1.6 Optimization

Keywords and phrases Valued CSP, Polymorphism, pp-definability, Geiger's Theorem

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.42

1 Introduction

The *constraint satisfaction problem* (CSP, for short) is a well-established framework for the uniform study of a wide range of both theoretical and applied problems. For the present purpose, it will be convenient to describe it in purely logical terms. A first-order sentence is *primitive positive* (pp, for short) if it is built up from atomic formulas with equality using only conjunction and existential quantifier. The CSP of a finite relational structure $\mathbf{B}$ asks to determine the pp-sentences valid in $\mathbf{B}$, in symbols

$$\mathrm{CSP}(\mathbf{B}) \coloneqq \{\varphi \mid \varphi \text{ is pp-sentence and } \mathbf{B} \vDash \varphi\}.$$

It is well known that $\mathrm{CSP}(\mathbf{B})$ can be identified with the set of *finite* structures $\mathbf{A}$ for which there is a homomorphism $\mathbf{A} \to \mathbf{B}$.

* The work was partly supported by the grant No. GA17-04630S of the Czech Science Foundation and partly by the long-term strategic development financing of the Institute of Computer Science (RVO:67985807).

The CSP problem of a finite relational structure is clearly decidable and, more precisely, belongs to the complexity class NP. The most outstanding problem in the field, known as the *dichotomy conjecture*, asks whether it is true that for every finite relational structure $\mathbf{B}$, either CSP($\mathbf{B}$) is NP-complete or it is tractable, i.e., solvable in polynomial time [17]. The algebraic approach to CSP has revealed very fruitful in this direction, leading to the discovery of striking connections with the theory of Maltsev conditions in universal algebra (see e.g. [11, 10, 19, 1, 2, 26]).

▶ **Example 1.** For any natural number n, a graph is *n-colorable* if it is possible to assign colors $\{0, \dots, n-1\}$ to its vertices in such a way that adjacent vertices receive different colors. We denote by $\mathbf{K}_n$ the complete n-element graph. It is easy to see that the set of finite n-colorable graphs is exactly CSP($\mathbf{K}_n$). Moreover, the problem CSP($\mathbf{K}_n$) is NP-complete for $n \geq 3$, and tractable otherwise. ◀

In this paper we consider a weighted generalization of the classical CSP, known as *valued constraint satisfaction problem* (VCSP, for short) see [25, 13, 29]. Roughly speaking, weighted structures are generalizations of classical structures in which relations are allowed to take degrees of truth (or, equivalently, payoffs) into a suitable valuation structure. Accordingly, the VCSP problem of a finite *weighted* relational structure $\mathbf{B}$ asks to determine an *optimal* solution for any pp-sentence φ, that is a tuple $\vec{b} \in B$ such that for any other tuple $\vec{c} \in B$ the value of φ on $\vec{b}$ is better or equal than its value on $\vec{c}$ (see the next section for a precise definition). As in the classical case, a major challenge in VCSP is to classify finite weighted structures according to whether their VCSP is tractable (solvable in polynomial time) or NP-hard.

Our approach differs from the one found in the VCSP literature in two fundamental aspects. On the one hand, the significant majority of works [29, 12, 27, 23, 21] are confined to the algebraic approach for VCSP formulated over a specific valuation structure, namely the set of positive rationals (with or without infinity) equipped with addition. On the contrary, our logically inspired approach is amenable to provide a uniform treatment for the VCSP formulated over arbitrary valuation structures. In this paper we explore this possibility for the case of locally finite valuation structures.

On the other hand, in the VCSP literature pp-sentences are (implicitly) understood as first-order formulas built up from atomic formulas using only existential quantifier and a kind of *monoidal* non-classical conjunction. As a drawback the VCSP, when formulated in these terms, does not embrace the classical aspects of weighted structures that are related to the usage of the classical conjunction (as opposed to the monoidal one). In this paper we allow the presence of both classical and monoidal conjunctions in pp-formulas. Consequently, our VCSP framework allows to model a richer class of optimization problems.

▶ **Example 2.** Using only the monoidal conjunction, one can model in VCSP the MAX-3SAT problem asking to find, for a given 3-CNF formula, a valuation satisfying the maximum number of clauses. Allowing also the classical conjunction, one can model a robust version of MAX-3SAT, where the input can be considered as not fully specified and we optimize w.r.t. the worst case. More precisely, the input is a finite number of 3-CNF formulas $\varphi_1, \dots, \varphi_m$ built up from the variables V. The task is to find a valuation $e \in \{0,1\}^V$ which maximizes the minimum number of satisfied clauses in all φ_i's. In Example 5, we will show how this problem can be properly formalized in our setting. ◀

The content of paper can be outlined as follows. In the classical CSP, the algebraic approach to the study of the dichotomy has revealed to be very fruitful. It is based on

Geiger's result [18] (independently by [7, 8]) characterizing the pp-definable relations of a finite structure $\mathbf{B}$ by means of *polymorphisms*, i.e., the homomorphisms $f\colon \mathbf{B}^n \to \mathbf{B}$ for any n. Our first main contribution is a generalization of Geiger's result to the case of VCSP over locally finite valuation structures. This is achieved by introducing a new notion of polymorphism for weighted structures, which generalizes the classical one. It should be mentioned that the VCSP literature contains already some generalizations of Geiger's result in terms of the so-called *weighted or fractional polymorphisms* [12, 22, 29, 23], but all of them are incomparable to our version at least for two reasons. First, the known generalizations are necessarily confined to pp-formulas without classical conjunction. Second, strictly speaking, they are not able to capture pp-definability in the original language: they do it only in an expanded language, where the expansion preserves tractability of VCSP. On the other hand, our approach provides a generalization of classical Geiger's Theorem (characterizing pp-definable relations in terms of polymorphisms) to the setting of weighted structures which does not require any expansion of the language. For this reason our work is not only a logical contribution to the computational study of VCSP (in which expansions preserving tractability are indistinguishable), but also to the development of weighted model theory (where different languages determine different structures).

Further we show that the VCSP over locally finite valuation structures can be reduced to finitely many CSP's. This reduction to the classical setting has two major consequences. On the one hand, it implies that several *tractability criteria* related to the existence of certain polymorphisms, which encompass the ones obtained in [28], transfer from CSP to VCSP. On the other and, this reduction yields that the dichotomy holds for our VCSP provided the dichotomy for classical CSP holds.[1] Interestingly enough, our results hold in full generality, in the sense that they are not confined to VCSP formulated over a specific valuation structure.

2 Preliminaries

The most general approach towards the study of VCSP complexity has been formulated taking semirings as valuation structures [3]. Here we restrict the attention to the so-called valued constraints, i.e., linearly ordered valuation structures (see for instance [15, 13, 25]).

An *integral Abelian pomonoid* is a structure $\mathbf{L} = \langle L, \leq, \odot, 0, 1\rangle$ such that $\langle L, \leq\rangle$ is a poset with a minimum 0 and maximum 1, $\langle L, \odot, 1\rangle$ is an Abelian monoid, and for every $\alpha, \beta, \gamma \in L$, if $\alpha \leq \beta$, then $\alpha \odot \gamma \leq \beta \odot \gamma$. When $\mathbf{L}$ is linearly ordered it can be equipped naturally with a lattice structure $\langle L, \wedge, \vee\rangle$. From now on we will assume that $\mathbf{L}$ is a fixed but arbitrary linearly-ordered integral Abelian pomonoid, and use the term *valuation structures* to denote these algebras.[2] A valuation structure is said to be *locally finite* if all finite generated subalgebras are finite.

▶ **Example 3.** Let $[0, 1]$ be the real unit interval. We denote by $\cdot$ the real multiplication, and by $\odot$ the operation defined as $a \odot b := \max\{0, a + b - 1\}$ for every $a, b \in [0, 1]$. Then the following algebras are valuation structures

$$Ł = \langle [0,1], \wedge, \vee, \odot, 0, 1\rangle, \qquad \mathbf{G} = \langle [0,1], \wedge, \vee, \wedge, 0, 1\rangle, \qquad \mathbf{P} = \langle [0,1], \wedge, \vee, \cdot, 0, 1\rangle.$$

[1] Recently, the dichotomy conjecture for the classical CSP has been claimed to be proven in [24] and [9].
[2] The usual definition of a valuation structure is the dual one from the order-theoretic point of view, namely the monoid unit is a bottom element. Thus we interpret elements of valuation structures as pay-offs rather than costs as is usual.

For every natural $n \geq 2$, we denote by $Ł_n$ the subalgebra of Ł with finite universe $\{0, \frac{1}{n-1}, \ldots, \frac{n-1}{n-1}\}$. Moreover, let $\mathbb{Q}_\perp^-$ be the set of negative rational with a least point $-\infty$. Then $\mathbf{Q}_\perp^- = \langle \mathbb{Q}_\perp^-, \wedge, \vee, +, -\infty, 0\rangle$ is a valuation structure as well.

The valuation structures Ł, $Ł_n$ and $\mathbf{G}$ are locally finite, while $\mathbf{P}$ and $\mathbf{Q}_\perp^-$ are not. ◀

We formulate VCSP in purely logical terms, by adapting the definition of the classical CSP to the case where the underlying first-order logic is non-classical instead of classical. We start by defining a suitable syntax. A language $\mathcal{L}$ is a countable set of domain variables together with a finite set of relational symbols (and their arities), binary propositional connectives $\odot, \wedge$ and the existential quantifier $\exists$. We call first-order formulas in this language *primitive positive formulas* or pp-formulas for short.[3] A pp-formula is called *pp-sentence* if it has no free variables, i.e., all its variables are under the scope of a quantifier. We further assume that every language contains a binary relational symbol $\approx$ denoting the classical equality.

Given a language $\mathcal{L}$ and a valuation structure $\mathbf{L}$, an $\mathbf{L}$-*structure or* $\mathbf{L}$-*template* for $\mathcal{L}$ is a tuple $\mathbf{B} = \langle B, \{R^{\mathbf{B}} \mid R \in \mathcal{L}\}\rangle$ such that B is a non-empty set, and for every relational symbol $R \in \mathcal{L}$ there is a map $R^{\mathbf{B}} \colon B^k \to L$ where k is the arity of R, i.e., $R^{\mathbf{B}}$ is a $\mathbf{L}$-valued (or, equivalently, weighted) relation assigning a cost/pay-off to any tuple $\vec{b} \in B^k$. [4] The relational symbol $\approx$ is always interpreted as the classical identity relation, i.e., $x \approx^{\mathbf{B}} y$ equals 1 if $x = y$ and 0 otherwise. When no confusion shall occur, we omit the superscripts from $R^{\mathbf{B}}$ and $\approx^{\mathbf{B}}$.[5] A $\mathbf{L}$-structure $\mathbf{B}$ is *finite* when B is finite. We assume throughout the paper that all $\mathbf{L}$-structures are finite.

A *valuation* in $\mathbf{B}$ is a mapping of the set of variables into B. The value $\|\varphi(\vec{b})\|^{\mathbf{B}}$ of a pp-formula $\varphi(\vec{x})$ under a valuation $\vec{x} \mapsto \vec{b}$ in the structure $\mathbf{B}$, is defined recursively as follows:

$$\begin{aligned}
\|R(\vec{b})\|^{\mathbf{B}} &= R^{\mathbf{B}}(\vec{b}),\\
\|\varphi_1(\vec{b_1}) \odot \varphi_2(\vec{b_2})\|^{\mathbf{B}} &= \|\varphi_1(\vec{b_1})\|^{\mathbf{B}} \odot \|\varphi_2(\vec{b_2})\|^{\mathbf{B}}\\
\|\varphi_1(\vec{b_1}) \wedge \varphi_2(\vec{b_2})\|^{\mathbf{B}} &= \|\varphi_1(\vec{b_1})\|^{\mathbf{B}} \wedge \|\varphi_2(\vec{b_2})\|^{\mathbf{B}}\\
\|\exists x \varphi(\vec{b}, x)\|^{\mathbf{B}} &= \bigvee_{a \in B} \|\varphi(\vec{b}, a)\|^{\mathbf{B}}.
\end{aligned}$$

The above join is always defined since $\mathbf{B}$ is finite. Moreover, since $\mathbf{L}$ is linearly ordered, the value of $\exists x \varphi(\vec{b}, x)$ is always witnessed, i.e., there is $a \in B$ such that $\|\exists x \varphi(\vec{b}, x)\|^{\mathbf{B}} = \|\varphi(\vec{b}, a)\|^{\mathbf{B}}$. Observe that every valuation structure satisfies the following distributive laws:

$$x \odot (y \vee z) = (x \odot y) \vee (x \odot z), \quad x \odot (y \wedge z) = (x \odot y) \wedge (x \odot z), \quad x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

Due to the above laws, pp-formulas have a *canonical normal form* (NF, for short):

$$\exists \vec{x} \bigwedge_{i \in I} \Big(\bigodot_{j \in J_i} \varphi_j \Big) \text{ where } \varphi_j \text{ is atomic.}$$

Observe that the value of a pp-sentence $\exists \vec{x} \varphi(\vec{x})$ in $\mathbf{B}$ is the same under any evaluation, and that it coincides with the maximum value that can be taken by the formula $\varphi(\vec{x})$ in $\mathbf{B}$. Thus,

---

[3] For the sake of simplicity, we give the same name to the propositional connectives of the language and the operations of the valuation structure.

[4] The notion of $\mathbf{L}$-structure corresponds exactly to the notion of *valued constraint language* used in the VCSP literature. Thus an $\mathbf{L}$-structure together with a pp-sentence corresponds to an instance of VCSP (cf. [13]).

[5] Observe that we assume that the identity relation $\approx$ is one of the basic relations R. For this reason, every time we state a definition imposing some condition on the basic relations R, we are assuming that the same condition holds for $\approx$.

we define a *solution* of a pp-sentence in NF $\exists\vec{x}\varphi(\vec{x})$ as any valuation $\vec{x} \mapsto \vec{b}$ such that

$$\|\varphi(\vec{b})\|^{\mathbf{B}} = \|\exists\vec{x}\varphi(\vec{x})\|^{\mathbf{B}}.$$

▶ **Definition 4.** Let $\mathbf{L}$ be a valuation structure and $\mathbf{B}$ an $\mathbf{L}$-structure. The *valued constraint satisfaction problem* for the template $\mathbf{B}$, in symbols VCSP($\mathbf{B}$), is the problem asking to find a solution to a given pp-sentence in NF.

▶ **Example 5.** We are now ready to formulate the robust version of MAX-3SAT problem mentioned in the introduction. Let $0 < \alpha < 1$ and let $\mathbf{B}$ be a $\mathbf{P}$-structure whose domain is $\{0,1\}$ endowed with relations $R_{\langle i,j,k\rangle}\colon \{0,1\}^3 \to [0,1]$ for each $\langle i,j,k\rangle \in \{0,1\}^3$, interpreted by letting $R_{\langle i,j,k\rangle}(x,y,z) = \alpha$ if $\langle x,y,z\rangle = \langle i,j,k\rangle$ and equal to 1 otherwise. Given for instance as input two 3-CNF formulas

$$\varphi_1 = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg z \vee u), \qquad \varphi_2 = (u \vee y \vee z) \wedge (\neg x \vee z \vee u),$$

we can encode it into the following pp-sentence:

$$\exists x,y,z,u\,(R_{\langle 0,1,0\rangle}(x,y,z) \odot R_{\langle 1,1,0\rangle}(x,z,u)) \wedge (R_{\langle 0,0,0\rangle}(u,y,z) \odot R_{\langle 1,0,0\rangle}(x,z,u)).$$

Observe this robust version of MAX-3SAT problem cannot be expressed in the VCSP formulated without classical conjunction. ◀

Given a finite $\mathbf{L}$-template $\mathbf{B}$, we say that VCSP($\mathbf{B}$) is *tractable* whenever it can be solved by an algorithm running in polynomial time in the length of input pp-sentence. Intractability and NP-hardness for VCSP($\mathbf{B}$) are defined analogously. It should be observed that in this definition, we assume that the computation in the valuation structure is done by an oracle and, therefore, does not affect the length of the computation.[6]

We say that a map (or, equivalently, a weighted relation) $R\colon B^k \to L$ is *pp-definable* in $\mathbf{B}$ when there is some pp-formula $\varphi(\vec{x})$ such that $R(\vec{b}) = \|\varphi(\vec{b})\|^{\mathbf{B}}$ for all $\vec{b} \in B^k$. It follows from the definition of VCSP that expanding the language of an $\mathbf{L}$-structure $\mathbf{B}$ with (finitely many) relations pp-definable in it does not change the tractability/intractability of VCSP($\mathbf{B}$).

Along the following sections, we will write $\mathbf{B}$ to denote an arbitrary $\mathbf{L}$-structure, for some valuation structure $\mathbf{L}$.

3 Polymorphisms and pp-definability characterization

Recall that the complexity of CSP($\mathbf{K}$), for a classical finite structure $\mathbf{K}$, depends only on the pp-definable relations in $\mathbf{K}$. These relations can be characterized in terms of polymorphisms as follows. An n-ary *polymorphism* (see e.g. [11]) of the structure $\mathbf{K}$ is an homomorphism $f\colon \mathbf{K}^n \to \mathbf{K}$ where $\mathbf{K}^n$ denotes the direct product of n many copies of $\mathbf{K}$. We denote by Pol($\mathbf{K}$) the clone of all polymorphisms of $\mathbf{K}$. The algebraic approach to the classical CSP is based on the following fundamental result [18]:

▶ **Geiger's Theorem 6.** *A relation R on K is pp-definable in $\mathbf{K}$ if and only if* $\mathrm{Pol}(\mathbf{K}) \subseteq \mathrm{Pol}(K,R)$*, where* $\mathrm{Pol}(K,R)$ *is the set of polymorphisms of the structure having the domain K and the only relation R.*

[6] Observe that, for the sake of simplicity, we defined VCSP($\mathbf{B}$) only for the case where the language of $\mathbf{B}$ is finite. However, the definitions of VCSP($\mathbf{B}$), tractability and NP-hardness can be reformulated in a way that embraces also the case where the language of $\mathbf{B}$ is infinite (see for instance [29, Section 1.3]). Our results, when suitably reformulated, remain true even if the assumption of the finiteness of the language is dropped.

We devote this section to a generalization of the above result to the case of VCSP, i.e., we characterize pp-definable relation in a finite $\mathbf{L}$-structure $\mathbf{B}$ as those which are preserved by suitably defined polymorphisms. The presented proof works for all locally finite valuation structures $\mathbf{L}$.

As in the classical case, our notion of a polymorphism shall depend on the concepts of homomorphisms and direct powers of $\mathbf{L}$-structures. Since $\mathbf{L}$-structures are in fact two-sorted structures, their natural homomorphisms are pairs of maps. The first acting on domains and the second on valuation structures. Moreover, these maps have to satisfy some additional properties ensuring that homomorphisms preserves values of pp-formulas.

▶ **Definition 7.** Let $\mathbf{L}, \mathbf{K}$ be valuation structures, $\mathbf{A}$ an $\mathbf{L}$-structure and $\mathbf{B}$ a $\mathbf{K}$-structure for the same language. A tuple $\langle f, \tau \rangle$ of maps $f\colon A \to B$ and $\tau\colon L \to K$ is a *homomorphism* if for all $\alpha, \beta \in L$, every k-ary relational symbols R and every $\vec{a} \in A^k$ we have

$$\tau(\alpha \odot \beta) \leq \tau(\alpha) \odot \tau(\beta), \quad \tau(\alpha \wedge \beta) \leq \tau(\alpha) \wedge \tau(\beta), \quad \tau(R^{\mathbf{A}}(\vec{a})) \leq R^{\mathbf{B}}(f(\vec{a})).$$

In the notation $f(\vec{a})$ the map f is applied component-wise.

It is worth to observe that if $\langle f, \tau \rangle$ is a homomorphism as above, then τ is a lattice homomorphism. This is a consequence of the fact that τ is a monotone map between chains.

As we mentioned, this notion of homomorphism preserves values of pp-formulas (modulo τ), in the sense that for every pp-formula $\varphi(\vec{x})$ and every tuple $\vec{a} \in A$, we have

$$\tau(\|\varphi(\vec{a})\|^{\mathbf{A}}) \leq \|\varphi(f(\vec{a}))\|^{\mathbf{B}}. \tag{1}$$

This fact can be easily proved by induction on the construction of φ.[7] If φ is a pp-sentence $\exists \vec{x} \psi(\vec{x})$, then (1) specializes to the following:

$$\tau(\|\exists \vec{x} \varphi(\vec{x})\|^{\mathbf{A}}) = \tau(\|\varphi(\vec{a})\|^{\mathbf{A}}) \leq \|\varphi(f(\vec{a}))\|^{\mathbf{B}} \leq \|\exists \vec{x} \varphi(\vec{x})\|^{\mathbf{B}},$$

where $\vec{a}$ is any tuple of elements from A witnessing the value $\|\exists \vec{x} \varphi(\vec{x})\|^{\mathbf{A}}$.

Direct powers of $\mathbf{L}$-structures are defined in the same way as direct powers of two-sorted structures. In particular, we form a direct power independently on both sorts, i.e., on domains and valuation structures. More precisely, the k-ary relations on the direct power $\mathbf{B}^n$ of n many copies of a $\mathbf{L}$-structure $\mathbf{B}$ are functions mapping k-tuples of elements from B^n to L^n. It may be useful to visualize the k-tuples of elements of B^n as matrices as follows:

$$\mathbb{A} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kn} \end{pmatrix}.$$

▶ **Definition 8.** Let $\mathbf{L}$ be valuation structures, $\mathbf{B}$ an $\mathbf{L}$-structure and $n \in \omega$. The direct power of n many copies of $\mathbf{B}$ is the $\mathbf{L}^n$-structure[8] $\mathbf{B}^n$ whose domain is B^n and for a k-ary relational symbol its interpretation $R^{\mathbf{B}^n}\colon B^{k \times n} \to L^n$ is defined for every matrix $\mathbb{A} \in B^{k \times n}$ having columns $\vec{a}_1, \dots, \vec{a}_n$ as follows:

$$R^{\mathbf{B}^n}(\mathbb{A}) \coloneqq \langle R^{\mathbf{B}}(\vec{a}_1), \dots, R^{\mathbf{B}}(\vec{a}_n) \rangle.$$

[7] The only non-trivial step of the induction is the one related to the existential quantifier, which requires to apply the observation that $\tau(x \vee y) = \tau(x) \vee \tau(y)$.

[8] Strictly speaking, $\mathbf{L}^n$ is not a valuation structure because it is not linearly ordered. Nevertheless one can introduce $\mathbf{L}^n$-structures analogously as $\mathbf{L}$-structures.

Given a map $f\colon B^n \to B$, we can extend it to a map from $B^{k\times n}$ to B^k by applying f row-wise. Thus, for a matrix $\mathbb{A} \in B^{k\times n}$ whose rows are $\vec{a}_1, \ldots, \vec{a}_k$ we define $f(\mathbb{A}) \coloneqq \langle f(\vec{a}_1), \ldots, f(\vec{a}_k)\rangle$.

▶ **Definition 9.** Let $\mathbf{B}$ be a $\mathbf{L}$-structure and $n \in \omega$. An n-ary *polymorphism* of $\mathbf{B}$ is a homomorphism from $\mathbf{B}^n$ to $\mathbf{B}$. The set of all polymorphisms of $\mathbf{B}$ is denoted $\mathrm{Pol}(\mathbf{B})$.

In other words, an n-ary polymorphism of $\mathbf{B}$ is a pair of maps $\langle f, \tau\rangle$ with $f\colon B^n \to B$ and $\tau\colon L^n \to L$ satisfying the following conditions:

1. For all $\vec{\alpha}, \vec{\beta} \in L^n$, it holds that $\tau(\vec{\alpha} \wedge \vec{\beta}) \leq \tau(\vec{\alpha}) \wedge \tau(\vec{\beta})$ and $\tau(\vec{\alpha} \odot \vec{\beta}) \leq \tau(\vec{\alpha}) \odot \tau(\vec{\beta})$.
2. For each k-ary relation $R^{\mathbf{B}}$ in $\mathbf{B}$ and matrix $\mathbb{A} \in B^{k\times n}$ we have $\tau R^{\mathbf{B}^n}(\mathbb{A}) \leq R^{\mathbf{B}}(f(\mathbb{A}))$.

The main result of this section is the following generalization of Geiger's Theorem:

▶ **Theorem 10.** *Let $\mathbf{B}$ be a finite $\mathbf{L}$-structure over a locally finite valuation structure $\mathbf{L}$. A relation $R\colon B^k \to L$ is pp-definable in $\mathbf{B}$ if, and only if, $\mathrm{Pol}(\mathbf{B}) \subseteq \mathrm{Pol}(B, R)$, where $\mathrm{Pol}(B, R)$ is the set of polymorphisms of the $\mathbf{L}$-structure having the domain B and the only relation R.*

▶ **Remark 11.** *It is worth noticing that valuation structures are subreducts of so-called MTL-algebras, the algebraic counterpart of the monoidal t-norm based logic MTL [16]. Consequently the above theorem in fact characterizes pp-definable relations in structures over locally finite MTL-chains.* ◀

Instead of presenting the technical details of the proof of Theorem 10, which are contained in the Appendix, let us describe the main ideas behind it.

First, we strongly rely on a correspondence between the components of our syntax, i.e., $\approx, \exists, \wedge, \odot$, and closure properties of the set of pp-definable relations in $\mathbf{B}$. More precisely, *permutations* of arguments and their *identifications* in a relation correspond to the presence of equality in the language. Meanwhile, the presence of existential quantifier is reflected in the fact that the pp-definable relations of $\mathbf{B}$ are closed under the following generalization of *projection*: for every relation $R\colon B^k \to L$, and $i \leq k$, the i-th projection of R is the $(k-1)$-ary relation $\pi_i[R]$, defined for every $\langle b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_k\rangle \in B^{k-1}$ as follows:

$$\pi_i[R](b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_k) \coloneqq \bigvee\{R(\vec{c}) \mid \vec{c} \in B^k,\ b_j = c_j \text{ for every } j \neq i\}.$$

Finally, the presence of $\odot$ and $\wedge$ in the syntax corresponds the fact that the pp-definable relations of $\mathbf{B}$ are closed under the formation of a certain generalization of *intersections* and *Cartesian products*.

Second, we rely on the fact that the pp-definable relations of a given arity k form a closure system on the lattice $\mathbf{L}^{B^k}$. Let us explain briefly why this is the case. First observe that, since $\mathbf{L}$ is locally finite and the language $\mathbf{B}$ is finite, there are only finitely many pp-definable k-ary relations in $\mathbf{B}$. Then consider a relation $R\colon B^k \to L$. There are only finitely many pp-definable k-ary relations $R_1, \ldots, R_m$ such that $R \leq R_i$. It is easy to see that $S = R_1 \wedge \cdots \wedge R_m$ is the least pp-definable relation which extends R (in the sense $R(\vec{b}) \leq S(\vec{b})$ for every $\vec{b} \in B^k$). Observe that its existence strongly depends on the fact that we allow the presence of *classical conjunction* in our pp-formulas. We denote by $\gamma^{(k)}(\cdot)$ the closure operator that assigns to any relation $R\colon B^k \to L$ the least pp-definable relation $\gamma^{(k)}(R)$ extending R. When no confusion shall occur, we will omit the superscript in $\gamma^{(k)}$.

Building on the two observations above, we are able to prove the following fundamental result, whose proof is contained in the Appendix. For a given $k \geq 1$ we define a matrix $\mathbb{C}_k \in B^{k\times|B|^k}$ whose *columns* are all k-tuples from B^k listed in a fixed linear order.

▶ **Extension Lemma 12.** *Let* $\mathbf{B}$ *be a finite* $\mathbf{L}$*-structure,* $\vec{b} \in B^k$*,* $n = |B|^k$ *and* $R\colon B^k \to L$ *a* k*-ary relation. Then there is an* n*-ary polymorphism* $\langle f, \tau\rangle \in \mathrm{Pol}(\mathbf{B})$ *such that* $f(\mathbb{C}_k) = \vec{b}$ *and* $\tau R^{\mathbf{B}^n}(\mathbb{C}_k) = \gamma(R)(\vec{b})$*.*

We conclude by sketching the proof of the main result of the section.

Proof of Theorem 10. The left-to-right implications follows from the natural observation that pp-definable relations in $\mathbf{B}$ are preserved by all polymorphisms of $\mathbf{B}$. This can be easily checked by induction on the complexity of formulas.

The converse implication follows from the extension Lemma 12. Take $\vec{b} \in B^k$. We have to show that R is pp-definable, i.e., $\gamma(R)(\vec{b}) = R(\vec{b})$. Let $n = |B|^k$. By Lemma 12 there is an n-ary polymorphism $\langle f, \tau\rangle \in \mathrm{Pol}(\mathbf{B})$ such that $f(\mathbb{C}_k) = \vec{b}$ and $\tau R(\mathbb{C}_k) = \gamma(R)(\vec{b})$. Since we assume that R is preserved by all polymorphisms, we obtain $\gamma(R)(\vec{b}) = \tau R(\mathbb{C}_k) \leq Rf(\mathbb{C}_k) = R(\vec{b})$. Thus $R = \gamma(R)$, i.e., R is pp-definable. ◀

4 Reducing VCSP to classical CSP

In this section we show that if $\mathbf{B}$ is a finite $\mathbf{L}$-structure and $\mathbf{L}$ is locally finite, then $\mathrm{VCSP}(\mathbf{B})$ can be reduced to a finite number of classical CSPs. In particular, this yields the Dichotomy Conjecture holds for these VCSPs modulo the Dichotomy Conjecture for classical CSP.

Along this section, we work with a fixed but arbitrary finite $\mathbf{L}$-structure $\mathbf{B} = \langle B, R_1, \dots, R_m\rangle$ where $\mathbf{L}$ is locally finite. Consider the ranges of relations $R_1, \dots, R_m$. Their union is a finite set $X \subseteq L$, generating a finite subalgebra $\mathbf{L}'$ of $\mathbf{L}$. It is obvious that pp-definable relations of $\mathbf{B}$ can attain values only in L'. Consequently, we can assume w.l.o.g. that $\mathbf{L} = \mathbf{L}'$ and, therefore, that $\mathbf{L}$ itself is finite. Accordingly, let $|L| = n$ and define $r \coloneqq n^n$. We say that r is the *rank* of $\mathbf{L}$.

▶ **Lemma 13.** *Let* $\vec{\alpha} = \langle \alpha_i \in L \mid i \in I\rangle$ *be a sequence of elements indexed by a finite set* I *such that* $|I| \geq r$*. Then there is* $J \subseteq I$ *such that* $|J| = r$ *and* $\bigodot_{i\in I} \alpha_i = \bigodot_{i\in J} \alpha_i$*.*

Proof. For any $\alpha \in L$ we have $\alpha^{n+1} = \alpha^n$ because $|L| = n$ and $\alpha \geq \cdots \geq \alpha^n \geq \alpha^{n+1}$. Since $|I| \geq n^n$, the sequence $\vec{\alpha}$ contains elements $\alpha \in L$ occurring more than n times. Using $\alpha^{n+1} = \alpha^n$, we can omit a suitable number of them and still preserve the value of $\bigodot_{i\in I} \alpha_i$.

Equivalently, let K be the variety of Abelian monoids axiomatized by the equation $x^{n+1} = x^n$. Since $\mathbf{L}$ has n elements, its monoidal reduct belongs to K. Observe that the n-generated free Abelian monoid over K has exactly n^n elements. Thus every product of elements in $\mathbf{L}$ is equivalent to a product of n^n many of them. ◀

We will make use of two basic transformations on $\mathbf{B}$. A first useful transformation of $\mathbf{B}$ consists in an expansion of the language by means of a particular family of pp-definable relations. The *cardinality* of a multiset I, denoted by $|I|$, is the number of elements of I counting repetitions. We consider the expansion of the language of $\mathbf{B}$ with new relational symbols R_I for each multiset of elements in $\{1, \dots, m\}$ of cardinality $\leq r$. The arity of R_I is the sum of all arities of R_i, $i \in I$, i.e., $ar(R_I) \coloneqq \sum_{i\in I} ar(R_i)$.

We denote by $\widehat{\mathbf{B}}$ the *expansion* of $\mathbf{B}$ obtained by adding to $\mathbf{B}$ the previous collection of relations R_I's interpreted as $R_I^{\widehat{\mathbf{B}}}(\vec{x}_i \mid i \in I) \coloneqq \bigodot_{i\in I} R_i^{\mathbf{B}}(\vec{x}_i)$ for $I = \langle t_1, \dots, t_q\rangle$.

Observe that the language of $\widehat{\mathbf{B}}$ is finite and, therefore $\widehat{\mathbf{B}}$ is a $\mathbf{L}$-structure as well. Moreover, since the expansion $\widehat{\mathbf{B}}$ is obtained by adding only pp-definable relations, the complexity of $\mathrm{VCSP}(\mathbf{B})$ coincides with the one of $\mathrm{VCSP}(\widehat{\mathbf{B}})$.

From a pp-formula φ of $\mathbf{B}$ in NF, we will define recursively a new pp-formula $\widehat{\varphi}$ in the language of $\widehat{\mathbf{B}}$ as follows. First consider the case where $\varphi = R_{t_1}(\vec{x}_{t_1}) \odot \cdots \odot R_{t_q}(\vec{x}_{t_q})$. Let I be the multiset whose elements are $t_1, \ldots, t_q$ counting repetitions. We define

$$\widehat{\varphi} := \begin{cases} R_I(\vec{x}_{t_1}, \ldots, \vec{x}_{t_q}) & \text{if } |I| \leq r, \\ \bigwedge\{R_J(\vec{x}_j \mid j \in J) \mid J \text{ is a submultiset of } I \text{ s.t. } |J| = r\} & \text{otherwise.} \end{cases}$$

Moreover, we set $\widehat{\varphi_1 \wedge \varphi_2} := \widehat{\varphi}_1 \wedge \widehat{\varphi}_2$ and $\widehat{\exists x\, \varphi} := \exists x\, \widehat{\varphi}$.

Observe that the symbol $\odot$ does not appear in $\widehat{\varphi}$. Moreover, it is easy to see that, since the valuation structure $\mathbf{L}$ is fixed, $\widehat{\varphi}$ can be constructed out of φ in polynomial time in the length of φ.[9] On the other hand, while obtaining $\widehat{\varphi}$ for an arbitrary valuation structure $\mathbf{L}$ has a quite high computational cost, lower bounds and easier constructions are likely to exist when each particular structure is studied separately.

▶ **Lemma 14.** *For every pp-formula φ of $\mathbf{B}$ in NF and $\vec{b} \in B$ it holds $\|\varphi(\vec{b})\|^{\mathbf{B}} = \|\widehat{\varphi}(\vec{b})\|^{\widehat{\mathbf{B}}}$.*

Proof. According to the definition of the translation $\widehat{\ }$, it suffices to check the equality for pp-formulas of the form $\varphi = \bigodot_{i \in I} R_i(\vec{x}_i)$. Since the monoidal unit $1 \in L$ is the top element, we have that for every $\alpha, \beta \in L$ it holds that $\alpha \odot \beta \leq \alpha \wedge \beta$. This fact easily implies that $\|\varphi(\vec{b})\|^{\mathbf{B}} \leq \|\widehat{\varphi}(\vec{b})\|^{\widehat{\mathbf{B}}}$. The other inequality is a direct consequence of Lemma 13. ◀

A second family of transformations of a structure $\mathbf{B}$ is given by *classicalizations* of it. Given $\alpha \in L$, we define a classical (non-weighted) structure $\mathbf{B}_\alpha$ whose language coincides with the one of $\mathbf{B}$, and every k-ary basic relation R is interpreted as follows: for every $\vec{b} \in B^k$ we set $\mathbf{B}_\alpha \vDash R(\vec{b}) \iff R^{\mathbf{B}}(\vec{b}) \geq \alpha$. We can apply these classicalizations to the structure $\widehat{\mathbf{B}}$ defined before, obtaining the following useful result:

▶ **Lemma 15.** *Let $\alpha \in L$, φ a pp-formula of $\mathbf{B}$ in NF and $\vec{b} \in B$. The following are equivalent:*

1. $\widehat{\mathbf{B}}_\alpha \vDash \widehat{\varphi}(\vec{b})$.
2. $\|\widehat{\varphi}(\vec{b})\|^{\widehat{\mathbf{B}}} \geq \alpha$.
3. $\|\varphi(\vec{b})\|^{\mathbf{B}} \geq \alpha$.

Proof. The equivalence between 2. and 3. follows from Lemma 14. The equivalence between 1. and 2. can be proven easily by induction on the construction of formulas (recall that $\odot$ does not appear in pp-formulas of the form $\widehat{\varphi}$). ◀

The next result shows that VCSP($\mathbf{B}$) can be reduced to a finite number of classical CSPs.

▶ **Theorem 16.**

- VCSP($\mathbf{B}$) *is tractable iff* CSP($\widehat{\mathbf{B}}_\alpha$) *is tractable for every $\alpha \in L$.*
- VCSP($\mathbf{B}$) *is NP-hard iff* CSP($\widehat{\mathbf{B}}_\alpha$) *is NP-hard for some $\alpha \in L$.*

Proof. Given a pp-sentence φ of $\widehat{\mathbf{B}}$ in which $\odot$ does not occur, we denote by φ^* the formula obtained from φ replacing the relations from the expansion $\widehat{\mathbf{B}}$ with the corresponding formulas

[9] This can be proved by induction on the construction of φ. The only non-trivial case is the one where $\varphi = R_{t_1}(\vec{x}_{t_1}) \odot \cdots \odot R_{t_q}(\vec{x}_{t_1})$. Let I be the multiset whose elements are $t_1, \ldots, t_q$ counting repetitions. It is easy to see that the length of $\widehat{\varphi}$ is bounded above by the binomial coefficient $\binom{|I|}{r}$. Since $\binom{|I|}{r}$ is bounded above by $|\varphi|^r$, where $|\varphi|$ is the length of φ, we are done.

of $\mathbf{B}$.[10] Observe that both the length of φ^* and the time needed in its construction are bounded above polynomially in the length of φ.

From Lemma 15 it follows that for every $\alpha \in L$ it holds that $\widehat{\mathbf{B}}_\alpha \vDash \varphi \iff \|\varphi^*\|^{\mathbf{B}} \geq \alpha$. Together with the fact that φ^* can be constructed in polynomial time in the length of φ, the above display easily implies that:

- If $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$ is NP-hard for some $\alpha \in L$, then also $\mathrm{VCSP}(\mathbf{B})$ is NP-hard as well.
- If $\mathrm{VCSP}(\mathbf{B})$ is tractable, then so is $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$ for every $\alpha \in L$.

Conversely, consider any pp-sentence φ of $\mathbf{B}$. Then define $\beta \coloneqq \max\{\alpha \in L \mid \widehat{\mathbf{B}}_\alpha \vDash \widehat{\varphi}\}$. From Lemma 15 it follows that the following conditions are equivalent for every $\vec{b} \in B$:

1. $\vec{b}$ is a solution for φ in $\mathbf{B}$.
2. $\vec{b}$ witnesses the fact that $\widehat{\mathbf{B}}_\beta \vDash \widehat{\varphi}$.

Suppose $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$ is tractable for every $\alpha \in L$ and consider a pp-sentence φ of $\mathbf{B}$. Recall that $\widehat{\varphi}$ can be constructed in polynomial time in the length of φ. Moreover, β can be calculated in polynomial time in the length of $\widehat{\varphi}$ by the assumption on the tractability of each of the $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$ and that fact that $\mathbf{L}$ is finite. It is well known that if $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$ is tractable, then given a pp-sentence ψ, the problem of then determining whether $\widehat{\mathbf{B}}_\alpha \vDash \psi$, and in the positive case produce a tuple $\vec{b}$ witnessing $\widehat{\mathbf{B}}_\alpha \vDash \psi$, is a tractable problem too [14]. Building on this, we obtain a tuple $\vec{b}$ from B witnessing the fact that $\widehat{\mathbf{B}}_\beta \vDash \widehat{\varphi}$ in polynomial time in the length of $\widehat{\varphi}$. Hence we conclude that this tuple $\vec{b}$ can be constructed in polynomial time in the length of φ. Since **2.** implies **1.** we conclude that $\vec{b}$ is a solution for φ in $\mathbf{B}$. Hence $\mathrm{VCSP}(\mathbf{B})$ is tractable as desired. Similarly, if $\mathrm{VCSP}(\mathbf{B})$ is NP-hard, so is at least one $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$. ◀

▶ **Corollary 17.** *The dichotomy conjecture holds for the VCSP formulated over finite* $\mathbf{L}$*-structures with* $\mathbf{L}$ *locally finite if and only if it holds for the classical CSP over finite templates.*

▶ **Remark 18.** *Observe that Theorem 16 is formulated under the assumption that* $\mathbf{L}$ *is finite, but it holds equivalently under the assumption that* $\mathbf{L}$ *locally finite. For this reason, the result applies to the case where* $\mathbf{L}$ *is* Ł*,* $\mathrm{Ł}_n$ *or* $\mathbf{G}$ *(see Example 3). In this setting the VCSP is also known under the names of weighted CSP and fuzzy CSP (cf. [25]).* ◀

As we mentioned, our formulation of the VCSP allows the presence of classical conjunction in pp-formulas. It is natural to ask whether this expansion of the language (w.r.t. pp-formulas without $\wedge$) preserves the tractability of the VCSP. In the next example we show that this is not the case in general.

▶ **Example 19.** Let $\mathbf{K}$ be a classical structure for which $\mathrm{CSP}(\mathbf{K})$ is NP-hard. We define an $\mathrm{Ł}_3$-structure $\mathbf{B}$ in the same language of $\mathbf{K}$ as follows. For every basic k-ary relation R, we set $R^{\mathbf{B}}(\vec{x}) = 1/2$ if $\vec{x} \in R^{\mathbf{K}}$ and equal to 0 otherwise.

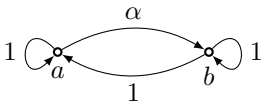
It is easy to see that the basic relations of $\widehat{\mathbf{B}}_{1/2}$ include the ones of $\mathbf{K}$. Hence $\mathrm{CSP}(\widehat{\mathbf{B}}_{1/2})$ is NP-hard. By Theorem 16 we conclude that $\mathrm{VCSP}(\mathbf{B})$ is NP-hard as well. On the other hand, when formulated without classical conjunction, the VCSP of $\mathbf{B}$ is tractable. To see this, let Γ be the set of pp-sentences without $\wedge$. It is easy to see that if $\varphi \in \Gamma$ contains at least two occurrences of a basic relation R different from the identity, then the $\|\varphi\|^{\mathbf{B}} = 0$ and, therefore, every tuple of elements of B is a solution of φ. Therefore the only meaningful pp-sentences have the following form (for some, possibly none, basic relation R): $\exists \vec{x}\, R(\vec{x}) \odot \bigodot_{i \in I}(y_i \approx z_i)$. It is not difficult to see that they can be handled in polynomial time. ◀

[10] If the classical conjunction $\wedge$ is not allowed in the language, this process cannot be done in general. See e.g. Example 19 below.

Even if the addition of classical conjunction to pp-sentences does not preserves tractability, the clone of polymorphisms $\mathrm{Pol}(\mathbf{B})$, which preserves $\wedge$, of an $\mathbf{L}$-structure $\mathbf{B}$ may contain useful information for handling pp-sentences in which $\wedge$ does not occur, as we remark in the next example.

▶ **Example 20.** Recall the $\langle s,t\rangle$-MIN-CUT problem. Given a weighted digraph $\mathbf{H} = \langle H, \mu, s, t\rangle$, where $\mu\colon H^2 \to \mathbb{N} \cup \{\infty\}$ is a map assigning weights to tuples of vertices ($\mu(u,v) = \infty$ if there is no arc between $u, v \in H$) and $s, t \in H$, this problem asks to find an $\langle s,t\rangle$-cut with a minimum weight. A $\langle s,t\rangle$-cut of $\mathbf{H}$ is a subset $C \subseteq H$ such that $s \in C$ and $t \notin C$. The weight of C is defined by $\sum_{u\in C, v\notin C} \mu(u,v)$.

The $\langle s,t\rangle$-MIN-CUT can be formulated within $\mathrm{VCSP}(\mathbf{B})$ over the $\mathbf{P}$-structure $\mathbf{B} = \langle \{a,b\}, R, P_s, P_t\rangle$ where $P_s(a) = 1$ and $P_s(b) = 0$, $P_t(b) = 1$ and $P_t(a) = 0$, and $R\colon \{a,b\}^2 \to [0,1]$ is defined in the following figure for $0 < \alpha < 1$.



The input weighted digraph can be encoded into a pp-sentence whose variables are vertices of $\mathbf{H}$ as follows:

$$\exists \vec{x}\, P_s(s) \odot P_t(t) \odot \bigodot \{R(x_i,x_j)^{\mu(x_i,x_j)} \colon x_i, x_j \in H \text{ and } \mu(x_i,x_j) \neq 0\}$$

where $R(x_i,x_j)^{\mu(x_i,x_j)} = R(x_i,x_j) \odot \ldots \odot R(x_i,x_j)$ repeated $\mu(x_i,x_j)$ many times.

It can be shown that $\mathrm{VCSP}(\mathbf{B})$ is intractable. This holds even if we replace the valuation structure $\mathbf{P}$ by a locally finite one provided that $\alpha^2 > 0$.[11] Nevertheless, the $\wedge$-less fragment is tractable since it admits a symmetric fractional polymorphism $\langle \min, \max\rangle$ of every arity (see [22]). Even though the full language version is intractable, the clone of polymorphisms $\mathrm{Pol}(\mathbf{B})$ still contains useful information. For instance $\langle \min, \tau\rangle$ and $\langle \max, \tau\rangle$ for $\tau(\alpha,\beta) = \alpha \odot \beta$ are polymorphisms of $\mathbf{B}$ where min and max are computed w.r.t. the linear order $a < b$. The fact that $\langle \min, \max\rangle$ is a fractional polymorphism can be captured by $\tau(R(\vec{x}), R(\vec{y})) \leq \tau(R(\min(\vec{x},\vec{y})), R(\max(\vec{x},\vec{y})))$ for all relations R which are pp-definable in the language $\exists, \odot$. Thus one can investigate fractional polymorphisms inside the clone $\mathrm{Pol}(\mathbf{B})$. ◀

5 Tractability conditions

In this section we will focus on tractability conditions for $\mathrm{VCSP}(\mathbf{B})$ consisting in existence of suitable polymorphisms of a finite $\mathbf{L}$-structure $\mathbf{B}$. Remarkably, given a polymorphism $\langle f, \tau\rangle \in \mathrm{Pol}(\mathbf{B})$, the component f is a classical polymorphism of the classical structure $\widehat{\mathbf{B}}_\alpha$ provided that $\alpha \leq \tau(\alpha, \ldots, \alpha)$. This will establish a connection between tractability conditions of the classical $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$ and tractability conditions for $\mathrm{VCSP}(\mathbf{B})$.

▶ **Definition 21.** Let $\alpha \in L$. A polymorphism $\langle f, \tau\rangle$ of $\mathbf{B}$ is *increasing in* α whenever $\alpha \leq \tau(\alpha, \ldots, \alpha)$. We will denote the set of polymorphisms of $\mathbf{B}$ increasing in α by $\mathrm{Pol}_{\uparrow\alpha}(\mathbf{B})$.

We say that a polymorphism is *locally increasing* if it is increasing in some value $\alpha \in L$. Such polymorphisms play a central role in the transfer of tractability conditions between $\mathrm{VCSP}(\mathbf{B})$ and the different $\mathrm{CSP}(\widehat{\mathbf{B}}_\alpha)$. More precisely, we have the following:

[11] This can be proved by showing non-existence of a ternary cyclic polymorphism in $\widehat{\mathbf{B}}_{\alpha^2}$ and then applying Theorem 16.

▶ **Lemma 22.** *We have* $\{f \mid \langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow\alpha}(\mathbf{B})\} \subseteq \mathrm{Pol}(\widehat{\mathbf{B}}_\alpha)$ *for all* $\alpha \in L$.

Proof. Let $\langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow\alpha}(\mathbf{B})$ be n-ary. Consider any k-ary relation R of $\widehat{\mathbf{B}}_\alpha$, and $\mathbb{B} \in B^{k\times n}$ such that $\mathbb{B} \in R$ (i.e., the columns of $\mathbb{B}$ belong to R). Observe that Lemma 15 implies that, for any $\vec{b} \in R$ it holds that $\alpha \le \|R^*(\vec{b})\|^{\mathbf{B}}$, where R^* is defined as in the proof of Lemma 16 (so that $\widehat{R^*} = R$). Then, by monotonicity of τ and the fact that $\langle f,\tau\rangle$ is increasing in α it follows that $\alpha \le \tau(\alpha,\dots,\alpha) \le \tau(\|R^*(\mathbb{B})\|^{\mathbf{B}})$, where $\|R^*(\mathbb{B})\|^{\mathbf{B}} = \langle \|R^*(\vec{b}_1)\|^{\mathbf{B}},\dots,\|R^*(\vec{b}_m)\|^{\mathbf{B}}\rangle$ for the columns $\vec{b}_1,\dots,\vec{b}_m$ of $\mathbb{B}$. Since $\langle f,\tau\rangle$ is a polymorphism of $\mathbf{B}$ it preserves pp-definable relations, and in particular R^*. Thus, $\tau(\|R^*(\mathbb{B})\|^{\mathbf{B}}) \le \|R^*(f(\mathbb{B}))\|^{\mathbf{B}}$. Lemma 15 implies that $f(\mathbb{B}) \in R$ concluding the proof. ◀

In order to provide a converse inclusion to the one from Lemma 22, we will construct for every classical polymorphism $f \in \mathrm{Pol}(\widehat{\mathbf{B}}_\alpha)$ a polymorphism $\langle f,\tau_f\rangle \in \mathrm{Pol}(\mathbf{B})$. To this end, recall that $\mathbb{C}_n$ is the matrix whose columns are all n-tuples from B^n, so its transposed $\mathbb{C}_n^T$ is the matrix whose rows are all the elements of B^n. Then for any $\vec{\alpha} \in L^n$ we let $R_{\vec{\alpha}}\colon B^{|B|^n} \to L$ be given by $R_{\vec{\alpha}}(\mathbb{C}_n^T) \coloneqq \vec{\alpha}$ and by $R_{\vec{\alpha}}(\vec{b}) \coloneqq 0$ for $\vec{b} \in B^{|B|^n} \setminus \mathrm{Columns}(\mathbb{C}_n^T)$.

Given a mapping $f\colon B^n \to B$, define $\tau_f\colon L^n \to L$ by $\tau_f(\vec{\alpha}) \coloneqq (\gamma(R_{\vec{\alpha}}))(f(\mathbb{C}_n^T))$ where γ is the closure operator defined in Section 3.

▶ **Lemma 23.** *We have* $\langle f,\tau_f\rangle \in \mathrm{Pol}_{\uparrow\alpha}(\mathbf{B})$ *iff* $f \in \mathrm{Pol}(\widehat{\mathbf{B}}_\alpha)$ *for any* $\alpha \in L$.

Proof. The left-to-right implication follows from Lemma 22. The converse implication follows from the Extension Lemma 12. The details are included in the Appendix. ◀

Sufficient tractability conditions for VCSP($\mathbf{B}$) arise now naturally from the ones for the associated CSP($\widehat{\mathbf{B}}_\alpha$).

▶ **Corollary 24.** *If for each* $\alpha \in L$ *there is* $\langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow\alpha}(\mathbf{B})$ *such that* f *satisfies a Maltsev condition implying tractability for classical CSP then* VCSP($\mathbf{B}$) *is tractable.*

Proof. Follows easily as a combination of Theorem 16 and Lemma 23. ◀

Of particular interest is the class of polymorphisms that are increasing in each value $\alpha \in L$. Precisely, we say that an n-ary polymorphism $\langle f,\tau\rangle$ of $\mathbf{B}$ is *fully increasing* (and write $\langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow}(\mathbf{B})$) whenever for each $\vec{\alpha} \in L^n$ it holds $\bigwedge_{1=i}^{n} \alpha_i \le \tau(\vec{\alpha})$.

It is easy to see that for an n-ary polymorphism $\langle f,\tau\rangle$ over an $\mathbf{L}$-structure it holds that

$$\langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow\alpha}(\mathbf{B}) \text{ for all } \alpha \in L \qquad \text{if and only if} \qquad \langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow}(\mathbf{B})$$

▶ **Lemma 25.** *The following conditions are equivalent:*
1. $f \in \mathrm{Pol}(\widehat{\mathbf{B}}_\alpha)$ *for all* $\alpha \in L$,
2. *There is some* τ *such that* $\langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow\alpha}(\mathbf{B})$ *for all* $\alpha \in L$,
3. *There is some* τ *such that* $\langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow}(\mathbf{B})$.

Proof. Clearly 3.⇒2., and 2.⇒3. follows easily by contraposition exploiting the monotonicity of τ. 2.⇒1. is a particular application of Lemma 22. Finally, 1.⇒3. follows by considering the mapping τ_f from Lemma 23, which is now increasing for each $\alpha \in L$. ◀

It is immediate that we can reformulate Corollary 24 in the particular case of fully increasing polymorphisms.

▶ **Corollary 26.** *If there is* $\langle f,\tau\rangle \in \mathrm{Pol}_{\uparrow}(\mathbf{B})$ *such that* f *satisfies a Maltsev condition implying tractability for classical CSP then* VCSP($\mathbf{B}$) *is tractable.*

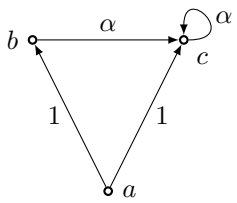


Figure 1 The relation R; the arcs with weight 0 are omitted.

The following example contains an application of fully increasing polymorphisms.

▶ **Example 27.** Let $\mathbf{L}$ be a valuation structure and $\alpha \in L$ such that $\alpha < 1$. Consider an $\mathbf{L}$-structure $\mathbf{B} = \langle \{a, b, c\}, R \rangle$ such that $R\colon B^2 \to L$ is a binary relation depicted in Figure 1. The $\mathbf{L}$-structure $\mathbf{B}$ has a fully-increasing polymorphism $\langle f, \tau \rangle$. If we introduce an order on B by setting $a < b < c$, then $f\colon B^2 \to B$ is given by $f(x, y) = \min\{x, y\}$ and $\tau(x, y) = x \vee y$ if $x, y > 0$ and equal to 0 otherwise. It is easy to check that $\langle f, \tau \rangle \in \mathrm{Pol}_{\uparrow}(\mathbf{B})$. Consequently, if $\mathbf{L}$ is locally finite, then VCSP($\mathbf{B}$) is tractable (Corollary 26) because f is a semilattice operation.

The problem VCSP($\mathbf{B}$) can have different interpretations depending also on the valuation structure $\mathbf{L}$. We describe one of them for $\mathbf{L} = \mathbf{Ł}_{n+2}$ for $n \geq 1$. The input for the problem is a finite number of digraphs $\mathbf{C}_1, \ldots, \mathbf{C}_m$. The task is to partition the union of vertices $\bigcup_{i=1}^{m} C_i$ into three parts denoted respectively a, b, c such that the following conditions are satisfied for all $i \in \{1, \ldots, m\}$:

1. $\mathbf{C}_i$ can have arcs coming from the part a to the parts b or c,
2. $\mathbf{C}_i$ can have at most n many arcs coming from the parts b or c to the part c,
3. no other arcs are allowed in $\mathbf{C}_i$.

If the above condition can be satisfied by some partitions, we are looking among them for a partition which minimizes the numbers of arcs coming from the parts b or c to the part c in the digraph $\mathbf{C}_i$ having the maximum number of such arcs. ◀

In the following example we show that it might be the case that for every $\alpha \in L$ there is a polymorphism f_α of the classical structure $\widehat{\mathbf{B}}_\alpha$ satisfying a fixed Maltsev condition C (e.g. majority), but there is no fully increasing polymorphism $\langle f, \tau \rangle \in \mathrm{Pol}_{\uparrow}(\mathbf{B})$ such that f satisfies C.

▶ **Example 28.** Let $\mathbf{A}$ be the $\mathbf{Ł_3}$-structure with universe $\{a, b, c\}$ and four binary relations:

P	a	b	c
a	1/2	0	1/2
b	0	0	0
c	0	1/2	0

Q	a	b	c
a	1/2	1/2	0
b	0	0	1/2
c	0	0	0

M	a	b	c
a	1/2	1/2	1/2
b	1	1	1/2
c	1/2	1/2	1

N	a	b	c
a	1	1/2	1/2
b	1/2	1	1
c	1/2	1/2	1/2

M, N are the only non empty relations in $\widehat{\mathbf{A}}_1$ and P, Q are the only non total ones in $\widehat{\mathbf{A}}_{1/2}$.

It is easy to check that a ternary majority function that maps $\langle x, y, z \rangle \mapsto b$ whenever x, y, z are different is a majority polymorphism in $\widehat{\mathbf{A}}_1$. Similarly, a majority function that maps $\langle x, y, z \rangle \mapsto a$ whenever x, y, z are different is a majority polymorphism in $\widehat{\mathbf{A}}_{1/2}$.

On the other hand, by simple calculations we can prove that any majority polymorphism f of $\widehat{\mathbf{A}}_1$ necessarily satisfies $f(a, b, c) = b$, and similarly, any majority polymorphism f' of $\widehat{\mathbf{A}}_{1/2}$ necessarily satisfies $f'(a, b, c) = a$. These two incompatible conditions, together with Lemma 25 make it impossible for $\mathbf{A}$ to have an increasing majority polymorphism. ◀

6 Conclusions

In this work we made first steps in the logical and algebraic study of VCSP over an arbitrary valuation structure, obtaining new connections with the classical CSP. This *logical* approach opens the door to several interesting problems. Among them we count the following:

- As model theory turned out to be a fundamental tool in the study of CSP over *infinite* templates [6, 4], we believe that model theory of first-order non-classical logic could shed some light on the analogous generalization of VCSP.
- Drawing our inspiration from [20, 5], we hope to generalize Datalog programs to our setting. In addition, this could help to prove that it is decidable whether Datalog programs can solve VCSP($\mathbf{B}$).
- Our version of Geiger's Theorem can be generalized at least in two directions. On the one hand, we wish to extend it to $\mathbf{L}$-structures where $\mathbf{L}$ is an arbitrary (possibly non-locally finite) valuation structure. On the other hand, we aim towards a version of Geiger's Theorem for pp-formulas without the classical conjunction $\wedge$ holding for arbitrary valuation structures.
- Last but not least, the logical perspective outlined in this paper could contribute to development of universal algebra and model theory of weighted structures. For instance, one might try to develop the notion of pp-interpretability which is crucial for the algebraic approach to the classical CSP.

References

1. L. Barto and M. Kozik. Absorbing Subalgebras, Cyclic Terms, and the Constraint Satisfaction Problem. *Logical Methods in Computer Science*, 8(1):1–26, 2012.
2. L. Barto and M. Kozik. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *Journal of the ACM*, 61(1):1–19, 2014.
3. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, 1997.
4. M. Bodirsky. Complexity classification in infinite-domain constraint satisfaction. *CoRR*, abs/1201.0856, 2012. URL: `http://arxiv.org/abs/1201.0856`.
5. M. Bodirsky and V. Dalmau. Datalog and constraint satisfaction with infinite templates. *Journal of Computer and System Sciences*, 79(1):79–100, 2013.
6. M. Bodirsky and J. Nešetřil. Constraint Satisfaction with Countable Homogeneous Templates. *Journal of Logic and Computation*, 16(3):359–373, 2006.
7. V. G. Bodnarchuk, L. A. Kaluzhnin, V. N. Kotov, and B. A. Romov. Galois theory for post algebras. I. *Cybernetics*, 5(3):243–252, 1969.
8. V. G. Bodnarchuk, L. A. Kaluzhnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. II. *Cybernetics*, 5(5):531–539, 1969.
9. A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. *CoRR*, abs/1703.03021, 2017. URL: `http://arxiv.org/abs/1703.03021`.
10. A. Bulatov and V. Dalmau. A Simple Algorithm for Mal'tsev Constraints. *SIAM Journal on Computing*, 36(1):16–27, 2006.
11. A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the Complexity of Constraints Using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.
12. D. Cohen, M. Cooper, P. Creed, P. Jeavons, and S. Živný. An Algebraic Theory of Complexity for Discrete Optimisation. *SIAM Journal on Computing*, 42(5):210–224, 2013.
13. D. Cohen, M. Cooper, P. G. Jeavons, and A. Krokhin. The complexity of soft constraint satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006.

14 D. A. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3):219–229, 2004.

15 M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2):199–227, 2004.

16 F. Esteva and L. Godo. Monoidal t-norm based logic: towards a logic for left-continuous t-norms. *Fuzzy Sets and Systems*, 124(3):271–288, 2001.

17 T. Feder and M. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

18 D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27(1), 1968.

19 P. Idziak, R. Mckenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. In *LICS*, pages 213–222, 2007.

20 P. Kolaitis and M. Vardi. Conjunctive-Query Containment and Constraint Satisfaction. *Journal of Computer and System Sciences*, 61(2):302–332, 2000.

21 V. Kolmogorov, A. Krokhin, and M. Rolinek. The Complexity of General-Valued CSPs. In *FOCS*, pages 1246–1258, 2015.

22 V. Kolmogorov, J. Thapper, and S. Živný. The Power of Linear Programming for General-Valued CSPs. *SIAM Journal on Computing*, 44(1):1–36, 2015.

23 M. Kozik and J. Ochremiak. Algebraic properties of valued constraint satisfaction problem. In *ICALP, Part I*, pages 846–858, 2015.

24 A. Rafiey, J. Kinne, and T. Feder. Dichotomy for digraph homomorphism problems. *CoRR*, abs/1701.02409, 2017. URL: `http://arxiv.org/abs/1701.02409`.

25 F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., 2006.

26 M. Siggers. A strong Mal'cev condition for locally finite varieties omitting the unary type. *Algebra Universalis*, 64(1-2):15–20, 2010.

27 J. Thapper and S. Živný. The complexity of finite-valued CSPs. In *STOC*, page 695. ACM Press, 2013.

28 E. Vodolazskii, B. Flach, and M. Schlesinger. Minimax problems of discrete optimization invariant under majority operators. *Computational Mathematics and Mathematical Physics*, 54(8):1327–1336, 2014.

29 S. Živný. *The Complexity of Valued Constraint Satisfaction Problems*. Cognitive Technologies. Springer Berlin Heidelberg, 2012.

A Proof of Lemma 12

Mimicking the classical formulation, given two k-ary relations $R, Q\colon B^k \to L$, we will write $R \subseteq Q$ whenever for each $\vec{a} \in B^k$ it holds that $R(\vec{a}) \leq Q(\vec{a})$. We will also identify classical crisp k-ary relations on B with maps from B^k to L whose range is $\{0, 1\}$. Given a k-ary relation $R\colon B^k \to L$, we define its *support* by $\mathrm{Supp}(R) = \{\vec{b} \in B^k \mid R(\vec{b}) \neq 0\}$. For a matrix $\mathbb{A} \in B^{k\times n}$ we denote the set of its columns by $\mathrm{Cols}(\mathbb{A})$.

The proof of Lemma 12 is based on properties of closure operators induced by pp-definable relations. Recall that for every arity k the system of pp-definable relations form a closure system. Thus the pp-definable relations induces a family of closure operators $\gamma^{(k)}$. We abuse the notation and denote all these closure operators simply γ because the arity of arguments fully determines the corresponding closure operator.

▶ **Lemma 29.** *Let $R, S\colon B^k \to L$ be relations. Then $\gamma(R \odot S) \subseteq \gamma(R) \odot \gamma(S)$.*

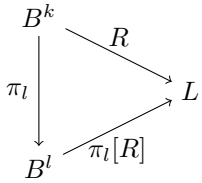


Figure 2 Projection.

Proof. We have $R \odot S \subseteq \gamma(R) \odot \gamma(S)$. Since the right-hand-side is pp-definable, this proves the claim. ◀

In what follows, we will show precisely how the transformations preserving pp-definable relations interact with the closure operators. There are three elementary transformations we are going to apply to relations, corresponding to existential quantification, permutation and identification of variables.

First, for $l < k$ there is a corresponding projection $\pi_l \colon B^k \to B^l$ (over the first l components). Given a relation $R \colon B^k \to L$, we define an l-ary relation $\pi_l[R] \colon B^l \to L$ by $\pi_l[R](\vec{a}) = \bigvee R[\pi_l^{-1}(\vec{a})]$.

Now we collect several properties of this transformation.

▶ **Lemma 30.** *Let $R, S \colon B^k \to L$, $T \colon B^l \to L$ and $\pi_l \colon B^k \to B^l$. Then*

1. *$R \subseteq S$ implies $\pi_l[R] \subseteq \pi_l[S]$.*
2. *$\pi_l[T\pi_l] = T$ where $T\pi_l$ is the composition of π_l followed by T (see Figure 3).*
3. *$R \subseteq \pi_l[R]\pi_l$. Moreover, for each $\vec{b} \in B^l$ there exists $\vec{b}' \in \pi_l^{-1}(\vec{b})$ such that $R(\vec{b}') = \pi_l[R](\vec{b})$.*
4. *Let $C \subseteq B^k$ be a crisp k-ary relation such that for each $\vec{a} \in B^l$ we have $C \cap \pi_l^{-1}(\vec{a}) \neq \emptyset$. Then $\pi_l[T\pi_l \odot C] = T$.*

Proof.

1. For $\vec{a} \in B^l$ we have by definition $\pi_l[R](\vec{a}) = \bigvee R[\pi_l^{-1}(\vec{a})] \leq \bigvee S[\pi_l^{-1}(\vec{a})] = \pi_l[S](\vec{a})$.
2. The composition $T\pi_l$ defines k-ary relation on B^k. Now pulling $T\pi_l$ along π_l gives $\pi_l[T\pi_l](\vec{a}) = \bigvee T\pi_l[\pi_l^{-1}(a)] = T(\vec{a})$ (see Figure 3).
3. Note that the composition of π_l followed by R approximates R from above, i.e., we have $R(\vec{a}) \leq \bigvee R[\pi_l^{-1}(\pi_l(\vec{a}))] = \pi_l[R](\pi_l(\vec{a}))$ for all $\vec{a} \in B^k$.
   Given $\vec{b} \in B^l$, consider $\vec{b}' \in \pi_l^{-1}(\vec{b})$ such that $R(\vec{b}')$ is maximum possible, i.e., $R(\vec{b}') = \bigvee R[\pi_l^{-1}(\vec{b})]$. Consequently, we have $\pi_l[R](\vec{b}) = \bigvee R[\pi_l^{-1}(\vec{b})] = R(\vec{b}')$.
4. We have $\pi_l[T\pi_l \odot C](\vec{a}) = \bigvee (T\pi_l \odot C)[\pi_l^{-1}(\vec{a})]$. Let $\vec{b} \in \pi_l^{-1}(\vec{a})$. If $C(\vec{b}) = 1$ then $(T\pi_l \odot C)(\vec{b}) = T\pi_l(\vec{b}) = T(\vec{a})$. If $C(\vec{b}) = 0$ then $(T\pi_l \odot C)(\vec{b}) = 0$. Since $C \cap \pi_l^{-1}(\vec{a}) \neq \emptyset$, we obtain $\pi_l[T\pi_l \odot C](\vec{a}) = T(\vec{a}) \vee 0 = T(\vec{a})$. ◀

Second, every permutation $\sigma \colon \{1, \ldots, k\} \to \{1, \ldots, k\}$ induces an automorphism $\sigma \colon B^k \to B^k$ by $\sigma(a_1, \ldots, a_k) = \langle a_{\sigma(1)}, \ldots, a_{\sigma(n)} \rangle$. Given a relation $R \colon B^k \to L$, we can define a relation $\sigma[R] = R\sigma^{-1}$.

▶ **Lemma 31.** *Let $R, S \colon B^k \to L$ and σ a permutation on k elements. Then*

1. *$R \subseteq S$ implies $\sigma[R] \subseteq \sigma[S]$,*
2. *$\sigma[R\sigma] = R\sigma\sigma^{-1} = R$,*
3. *$\sigma[R]\sigma = R\sigma^{-1}\sigma = R$.*

Proof. For the first claim we have $\sigma[R](\vec{a}) = R(\sigma^{-1}(\vec{a})) \leq S(\sigma^{-1}(\vec{a})) = \sigma[S](\vec{a})$. The rest is obvious. ◀

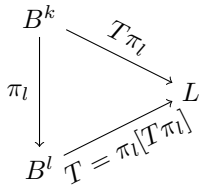


Figure 3 Projection identity.

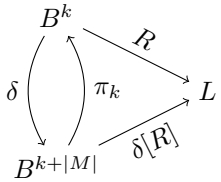


Figure 4 Duplicating.

Finally, we introduce a duplicating operator. Let $M\colon \{1,\dots,k\} \to \mathbb{N}$ be a multiset, i.e., a function assigning to every $i \in \{1,\dots,k\}$ its multiplicity. The cardinality of M is defined by $|M| = \sum_{i=1}^{k} M(i)$. Define $\delta\colon B^k \to B^{k+|M|}$ by

$$\delta(a_1,\dots,a_k) = \langle a_1,\dots,a_k,\underbrace{a_1,\dots,a_1}_{M(1)},\dots,\underbrace{a_k,\dots,a_k}_{M(k)}\rangle,$$

i.e., δ duplicates coordinates according to M.

Given a k-ary relation $R\colon B^k \to L$, we define a $k+|M|$-ary relation $\delta[R]\colon B^{k+|M|} \to L$ by

$$\delta[R] = R\pi_k \odot E,$$

where E is the crisp relation expressing that the first coordinate x_1 equals to coordinates $x_{k+1},\dots,x_{k+M(1)}$, the second coordinate x_2 to coordinates $x_{k+M(1)+1},\dots,x_{k+M(1)+M(2)}$ etc. Thus

$$E = \left(\bigodot_{i=1}^{M(1)} (x_1 \approx x_{k+i})\right) \odot \left(\bigodot_{i=M(1)+1}^{M(1)+M(2)} (x_2 \approx x_{k+i})\right) \odot \dots$$

Note that $\delta[R]$ is pp-definable if R is pp-definable via the crisp equality relation which we assume to have in the language.

As in the previous cases, we can show several useful properties of the duplicating operators.

▶ **Lemma 32.** *Let $R, S\colon B^k \to L$ and $T\colon B^{k+|M|} \to L$. Then*

1. *$R \subseteq S$ implies $\delta[R] \subseteq \delta[S]$.*
2. *$\delta[R] \subseteq E$.*
3. *$\delta[T\delta] \subseteq T$. Moreover, if $T \subseteq E$ then $\delta[T\delta] = T$.*
4. *$\delta[R]\delta = R$.*
5. *$\pi_k[\delta[R]] = R$.*
6. *$T \subseteq E$ implies $\delta[\pi_k[T]] = T$.*

Proof.

1. We have that $\delta[R](\vec{a}) = R(\pi_k(\vec{a})) \odot E(\vec{a}) \leq S(\pi_k(\vec{a})) \odot E(\vec{a}) = \delta[S](\vec{a})$.

2. Follows from the definition of δ.
3. We have $\delta[T\delta](\vec{a}) = T\delta(\pi_k(\vec{a})) \odot E(\vec{a}) \leq T(\vec{a})$. The last inequality holds because if $E(\vec{a}) \neq 0$ (i.e., $E(\vec{a}) = 1$) then $\delta\pi_k(\vec{a}) = \vec{a}$.
4. We have $\delta[R](\delta(\vec{a})) = R(\pi_k(\delta(\vec{a}))) \odot E(\delta(\vec{a})) = R(\vec{a})$.
5. We have $\pi_k[\delta[R]](\vec{a}) = \bigvee \delta[R][\pi_k^{-1}(\vec{a})]$. Let $\vec{b} \in \pi_k^{-1}(\vec{a})$. Then

$$\delta[R](\vec{b}) = R(\pi_k(\vec{b})) \odot E(\vec{b}) = \begin{cases} R(\vec{a}) & \text{if } E(\vec{b}) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

   In particular, for $\vec{b} = \delta(\vec{a}) \in \pi_k^{-1}(\vec{a}) \cap E$ we have $\delta[R](\delta(\vec{a})) = R(\vec{a})$. Thus $\pi_k[\delta[R]](\vec{a}) = R(\vec{a})$.
6. We have $\delta[\pi_k[T]](\vec{a}) = \pi_k[T](\pi_k(\vec{a})) \odot E(\vec{a})$. If $E(\vec{a}) = 0$ then $\delta[\pi_k[T]](\vec{a}) = 0$ and $T(\vec{a}) = 0$ as well because $T \subseteq E$. Suppose that $E(\vec{a}) = 1$. We have $\delta[\pi_k[T]](\vec{a}) = \pi_k[T](\pi_k(\vec{a})) = \bigvee T[\pi_k^{-1}\pi_k(\vec{a})] = T(\vec{a})$. The last equality follows from the fact that $\pi_k^{-1}\pi_k(\vec{a}) \cap E = \{\vec{a}\}$. ◀

Relying on the previously proven properties, we can show that the closure operator γ commutes with projections, permutations and duplications.

▶ **Lemma 33.** *Let $R\colon B^k \to L$. Then*
1. *For $l < k$ and $\pi_l\colon B^k \to B^l$ a projection it holds that $\gamma(\pi_l[R]) = \pi_l[\gamma(R)]$.*
2. *For $\sigma\colon B^k \to B^k$ a permutation it holds that $\gamma(\sigma[R]) = \sigma[\gamma(R)]$.*
3. *For $\delta\colon B^k \to B^{k+|M|}$ a duplication it holds that $\gamma(\delta[R]) = \delta[\gamma(R)]$.*

Proof.
1. Since $R \subseteq \gamma(R)$, we have $\pi_l[R] \subseteq \pi_l[\gamma(R)]$ by Lemma 30. Thus $\gamma(\pi_l[R]) \subseteq \pi_l[\gamma(R)]$ follows since pp-definable relations are closed under projections.
   Conversely, we have $R \subseteq \pi_l[R]\pi_l$ by Lemma 30. Further, $\pi_l[R]\pi_l \subseteq \gamma(\pi_l[R])\pi_l$. Since the right-hand side is pp-definable, we obtain $\gamma(R) \subseteq \gamma(\pi_l[R]\pi_l) \subseteq \gamma(\pi_l[R])\pi_l$. Applying π_l to both sides, we obtain $\pi_l[\gamma(R)] \subseteq \gamma(\pi_l[R])$ by Lemma 30.
2. We have $\sigma[R] \subseteq \sigma[\gamma(R)]$ by Lemma 31. Since γ-closed sets are closed under permutations of coordinates, $\gamma(\sigma[R]) \subseteq \sigma[\gamma(R)]$.
   Conversely, we have $\sigma[R] \subseteq \gamma(\sigma[R])$ by the properties of closure operators. By Lemma 31 we have $R = \sigma^{-1}[\sigma[R]] \subseteq \sigma^{-1}[\gamma(\sigma[R])]$. Thus also $\gamma(R) \subseteq \sigma^{-1}[\gamma\sigma[R])]$ since pp-definable relations are closed under permutations of coordinates. Consequently, by Lemma 31 $\sigma[\gamma(R)] \subseteq \sigma[\sigma^{-1}[\gamma(\sigma[R])]] = \gamma(\sigma[R])$.
3. Since $\delta[\gamma(R)]$ is pp-definable, we have $\gamma\delta[R] \subseteq \delta[\gamma(R)]$ using also Lemma 32.
   Conversely we have $\delta[R] \subseteq \gamma(\delta[R])$. Thus by Lemma 30 and 32 we get $R = \pi_k[\delta[R]] \subseteq \pi_k[\gamma(\delta[R])]$. Since the right-hand-side is pp-definable, $\gamma(R) \subseteq \pi_k[\gamma(\delta[R])]$ follows. Applying δ and using Lemma 32, we obtain $\delta[\gamma(R)] \subseteq \delta[\pi_k[\gamma(\delta[R])]]$. Since $\delta[R] \subseteq E$, we have $\gamma(\delta[R]) \subseteq E$ because E is pp-definable. Finally, using Lemma 32, we have $delta[\gamma(R)] \subseteq \delta[\pi_k[\gamma(\delta[R])]] = \gamma(\delta[R])$. ◀

Next we present a useful characterization of polymorphisms. Recall that we extent an n-ary map $f\colon B^n \to B$ so that it acts on matrices $\mathbb{A} \in B^{k\times n}$. Since f acts row-wise on $\mathbb{A}$, it commutes with the projections π_l, permutations σ and duplications δ. In particular, we have $\pi_l\sigma\delta f(\mathbb{A}) = f\pi_l\sigma\delta(\mathbb{A})$ for all $\mathbb{A} \in B^{k\times n}$.

For a given $k \geq 1$ we will define a matrix $\mathbb{C}_k \in B^{k\times|B|^k}$. The matrix $\mathbb{C}_k$ is the matrix whose *columns* are all k-tuples from B^k listed in a linear order, i.e., we have $\mathrm{Cols}(\mathbb{C}_k) = B^k$. Observe that $\mathbb{C}_k$ has pairwise distinct rows. Moreover, the transpose matrix $\mathbb{C}_k^T$ is the matrix whose *rows* are all k-tuples from B^k.

▶ **Lemma 34.** *Let* $f\colon B^n \to B$ *and* $\tau\colon L^n \to L$ *such that*

$$\tau(\vec{\alpha} \wedge \vec{\beta}) \leq \tau(\vec{\alpha}) \wedge \tau(\vec{\beta}), \qquad \tau(\vec{\alpha} \odot \vec{\beta}) \leq \tau(\vec{\alpha}) \odot \tau(\vec{\beta}).$$

The tuple $\langle f, \tau\rangle$ *is a polymorphism iff for all relations* $S\colon B^{|B|^n} \to L$ *we have*

$$\tau S(\mathbb{C}_n^T) \leq \gamma(S)(f(\mathbb{C}_n^T)).$$

Proof. The left-to-right direction follows since polymorphisms preserve all pp-definable relations. More precisely, since τ is monotone, we have $\tau S(\mathbb{C}_n^T) \leq \tau\gamma(S)(\mathbb{C}_n^T) \leq \gamma(S)(f(\mathbb{C}_n^T))$.

Conversely, suppose $\tau S(\mathbb{C}_n^T) \leq \gamma(S)(f(\mathbb{C}_n^T))$ for all $S\colon B^{|B|^n} \to L$. Let $R\colon B^k \to L$ be a k-ary relation in $\mathbf{B}$ and $\mathbb{A} \in B^{k\times n}$. We have to show that $\tau R(\mathbb{A}) \leq Rf(\mathbb{A})$.

Since the $\mathrm{Rows}(\mathbb{C}_n^T) = B^n$, there is a transformation $\pi_k\sigma\delta\colon B^{|B|^n} \to B^k$ such that $\mathbb{A} = \pi_k\sigma\delta(\mathbb{C}_n^T)$. In other words, applying a suitable duplicating operator followed by a permutation and then by a projection to the matrix $\mathbb{C}_n^T$, we can produce the matrix $\mathbb{A}$. Define $S\colon B^{|B|^n} \to L$ by the composition $R\pi_k\sigma\delta$. By the assumption $\tau S(\mathbb{C}_n^T) \leq \gamma(S)f(\mathbb{C}_n^T)$, Lemma 30, 31, 32 on properties of projections, permutations, duplicating operator and Lemma 33 on properties of the γ's operators, we have the following chain of (in)equalities that concludes the proof:

$$\begin{aligned}
\tau R(\mathbb{A}) &= \tau R\pi_k\sigma\delta(\mathbb{C}_n^T) = \tau S(\mathbb{C}_n^T) \leq \gamma(S)(f(\mathbb{C}_n^T)) = \delta[\gamma(S)](\delta f(\mathbb{C}_n^T)) = \gamma(\delta[S])(f\delta(\mathbb{C}_n^T))\\
&= \gamma(\delta[R\pi_k\sigma\delta])(f\delta(\mathbb{C}_n^T)) \leq \gamma(R\pi_k\sigma)(f\delta(\mathbb{C}_n^T)) = \sigma[\gamma(R\pi_k\sigma)](\sigma f\delta(\mathbb{C}_n^T))\\
&= \gamma(\sigma[R\pi_k\sigma])(f\sigma\delta(\mathbb{C}_n^T)) = \gamma(R\pi_k)(f\sigma\delta(\mathbb{C}_n^T)) \leq \pi_k[\gamma(R\pi_k)](\pi_k f\sigma\delta(\mathbb{C}_n^T))\\
&= \gamma(\pi_k[R\pi_k])(f\pi_k\sigma\delta(\mathbb{C}_n^T)) = \gamma(R)(f(\mathbb{A})) = Rf(\mathbb{A}).
\end{aligned}$$

◀

Let $\mathbb{A} \in B^{k\times n}$ and $\vec{b} \in B^k$. There is one-to-one correspondence between elements of L^n and relations $R\colon B^k \to L$ such that $\mathrm{Supp}(R) = \mathrm{Cols}(\mathbb{A})$. Given a tuple $\vec{\alpha} \in L^n$, the corresponding relation $R_{\vec{\alpha}}\colon B^k \to L$ is the relation whose support is $\mathrm{Cols}(\mathbb{A})$ and is defined by $R_{\vec{\alpha}}(\mathbb{A}) = \vec{\alpha}$.

▶ **Lemma 35.** *The map* $\tau\colon L^n \to L$ *defined by* $\tau(\vec{\alpha}) = \gamma(R_{\vec{\alpha}})(\vec{b})$ *satisfies*

$$\tau(\vec{\alpha} \wedge \vec{\beta}) \leq \tau(\vec{\alpha}) \wedge \tau(\vec{\beta}) \qquad \text{and} \qquad \tau(\vec{\alpha} \odot \vec{\beta}) \leq \tau(\vec{\alpha}) \odot \tau(\vec{\beta}).$$

Proof. Let $\vec{\alpha}, \vec{\beta} \in L^n$. The first inequality is equivalent to τ being monotone in all arguments. To see that τ is monotone, note that $\vec{\alpha} \leq \vec{\beta}$ implies $\gamma(R_{\vec{\alpha}}) \subseteq \gamma(R_{\vec{\beta}})$. Thus we have

$$\tau(\vec{\alpha}) = \gamma(R_{\vec{\alpha}})(\vec{b}) \leq \gamma(R_{\vec{\beta}})(\vec{b}) = \tau(\vec{\beta}).$$

Further, we have $R_{\vec{\alpha}\odot\vec{\beta}} = R_{\vec{\alpha}} \odot R_{\vec{\beta}}$. Using Lemma 29, we get

$$\tau(\vec{\alpha} \odot \vec{\beta}) = \gamma(R_{\vec{\alpha}} \odot R_{\vec{\beta}})(\vec{b}) \leq \gamma(R_{\vec{\alpha}})(\vec{b}) \odot \gamma(R_{\vec{\beta}})(\vec{b}) = \tau(\vec{\alpha}) \odot \tau(\vec{\beta}).$$ ◀

Now we are ready to finish the proof of the extension lemma.

Proof of Lemma 12. Let $n = |B|^k$. Note that the matrix $\mathbb{C}_k$ cannot contain a row more than once. Thus one can extend the matrix $\mathbb{C}_k$ by adding new rows to a matrix $\mathbb{D}_k \in B^{|B|^n\times n}$ so that the rows of $\mathbb{D}_k$ consists of all n-tuples from B^n and $\mathbb{C}_k = \pi_k(\mathbb{D}_k)$ where π_k is applied column-wise. Let $T\colon B^{|B|^n} \to L$ be a relation such that $T = R\pi_k \odot \mathrm{Cols}(\mathbb{D}_k)$. Note that $T(\mathbb{D}_k) = R(\mathbb{C}_k)$. By Lemma 30 it follows that $\pi_k[T] = R$. Indeed, for each $\vec{a} \in B^k$ (i.e., $\vec{a}$ is a j-th column of $\mathbb{C}_k$) there is a column $\vec{a}'$ in $\mathbb{D}_k$ such that $\pi_k(\vec{a}') = \vec{a}$. Thus we have $\mathrm{Cols}(\mathbb{D}_k) \cap \pi_k^{-1}(\vec{a}) \neq \emptyset$.

Further by Lemma 30 there exists $\vec{b}' \in \pi_k^{-1}(\vec{b})$ such that $\pi_k[\gamma(T)](\vec{b}) = \gamma(T)(\vec{b}')$. We define $f(\mathbb{D}_k) = \vec{b}'$. Further we define $\tau(\vec{\alpha}) = \gamma(R_{\vec{\alpha}})(\vec{b}')$ as in Lemma 35 for $\mathrm{Supp}(R_{\vec{\alpha}}) = \mathrm{Cols}(\mathbb{D}_k)$.

In order to prove that $\langle f, \tau \rangle$ is a polymorphism, it suffices to show by Lemma 34 that $\tau S(\mathbb{D}_k) \leq \gamma(S)(f(\mathbb{D}_k))$ for all relations $S \colon B^{|B|^n} \to L$.[12] Let $S \colon B^{|B|^n} \to L$. There is $\vec{\alpha} \in L^n$ such that $R_{\vec{\alpha}} = S \odot \mathrm{Cols}(\mathbb{D}_k)$. By definition of τ we have

$$\tau S(\mathbb{D}_k) = \tau R_{\vec{\alpha}}(\mathbb{D}_k) = \gamma(R_{\vec{\alpha}})(\vec{b}') \leq \gamma(S)(\vec{b}') = \gamma(S)(f(\mathbb{D}_k)).$$

Thus $\langle f, \tau \rangle$ is a polymorphism.

It remains to prove that $f(\mathbb{C}_k) = \vec{b}$ and $\tau R(\mathbb{C}_k) = \gamma(R)(\vec{b})$. To see the first equality, observe that

$$f(\mathbb{C}_k) = f(\pi_k(\mathbb{D}_k)) = \pi_k(f(\mathbb{D}_k)) = \pi_k(\vec{b}') = \vec{b}.$$

Regarding the second equality, since $\mathrm{Supp}(T) = \mathrm{Cols}(\mathbb{D}_k)$, we have $T = R_{\vec{\alpha}}$ for some $\vec{\alpha} \in L^n$. Consequently,

$$\tau R(\mathbb{C}_k) = \tau T(\mathbb{D}_k) = \gamma(T)(\vec{b}') = \pi_k[\gamma(T)](\vec{b}) = \gamma(\pi_k[T])(\vec{b}) = \gamma(R)(\vec{b}). \qquad \blacktriangleleft$$

B Proof of Lemma 23

Regarding the proof of the right-to-left implication, assume that $f \in \mathrm{Pol}(\widehat{\mathbf{B}}_\alpha)$ for all $\alpha \in L$. Observe that the relation $R_{\vec{\alpha}}$ is a particular case of the relations introduced before Lemma 35 (with support $\mathbb{C}_n^T$). Thus τ_f satisfies

$$\tau_f(\vec{\alpha} \wedge \vec{\beta}) \leq \tau_f(\vec{\alpha}) \wedge \tau_f(\vec{\beta}) \qquad \text{and} \qquad \tau_f(\vec{\alpha} \odot \vec{\beta}) \leq \tau_f(\vec{\alpha}) \odot \tau_f(\vec{\beta}),$$

as a consequence of Lemma 35. Moreover, following the same reasoning as in the proof of Lemma 12, we know it is a polymorphism of $\mathbf{B}$ (relying on Lemma 34).

It suffices then to show that τ_f is increasing in α. Let $\vec{\alpha} = \langle \alpha, \ldots, \alpha \rangle$ be the vector of length n. Since the relation $\gamma(R_{\vec{\alpha}})$ is characterized by a $|B|^n$-ary pp-formula $\varphi(\vec{x})$ in $\mathbf{B}$, $\widehat{\varphi}(\vec{x})$ is a pp-formula in $\widehat{\mathbf{B}}_\alpha$. By definition $\vec{\alpha} = R_{\vec{\alpha}}(\mathbb{C}_n^T) \leq \gamma(R_{\vec{\alpha}})(\mathbb{C}_n^T) = \|\varphi(\mathbb{C}_n^T)\|^{\mathbf{B}}$. Thus, by Lemma 15 it follows that the columns of $\mathbb{C}_n^T$ belong to the relation pp-defined by $\widehat{\varphi}(\vec{x})$ in the classical structure $\widehat{\mathbf{B}}_\alpha$.

Since $f \in \mathrm{Pol}(\widehat{\mathbf{B}}_\alpha)$, it preserves pp-definable formulas, in particular $\widehat{\varphi}(\vec{x})$. Thus $f(\mathbb{C}_n^T)$ belongs to the relation pp-defined by $\widehat{\varphi}(\vec{x})$ in $\widehat{\mathbf{B}}_\alpha$. Consequently, Lemma 15 implies

$$\alpha \leq \|\varphi(f(\mathbb{C}_n^T))\|^{\mathbf{B}} = \gamma(R_{\vec{\alpha}})(f(\mathbb{C}_n^T)) = \tau_f(\vec{\alpha})$$

which concludes the proof. $\blacktriangleleft$

[12] Observe that $\mathbb{C}_n^T = \mathbb{D}_k$.